Mono » OpenSUSE 10.2 » Тиманн » Firefox

LINUX FORMAT

Главное в мире Linux

Февраль 2007 № 2 (89)

LXF DVD внутри!



OpenSUSE 10.2

Плюс: Учебник по OpenOffice.org и более 50 крутых приложений

КОНКУРС !!! с.8

Он объединяет Windows .NET и Linux:

Mono уже с нами!

Самые убойные приложения 2007 года – на вашем рабочем столе:

- Beagle, Tomboy, Banshee
- Новые возможности
- Основы C#

mono

Десятка лучших расширений для Firefox

Список начинается на с. 34

Учитесь вместе с нами!

Сборка собственного ядра с. 70

Примитивы и кривые Безье в Blender с. 98

Java Enterprise Edition с. 82

Демоны Unix с. 78



«Я не хотел быть бизнесменом, я хотел быть программистом!»

Майкл Тиманн из Red Hat с.30

LINUX FORMAT
Главное в мире Linux

К Вашим услугам...

В фокусе этого номера Linux Format – приложения Mono, поэтому мы спросили у команды LXF: «Если бы Beagle мог искать что-то в вашей жизни, что бы это было?»



Пол Хадсон
Моя жена Илдико. Она все время теряется в магазинах одежды, и я не могу найти ее часами.



Грэм Моррисон
Я бы попросил Beagle поискать ответ на вопрос, почему я все время что-то ищу.



Майк Сондерс
Штуку, что запускается с любым приложением Gnome, сжирая всю память... А, да это же Evolution Data Server!



Эфрейн Эрнандес-Мендоса
Мою новую крошку. Я уверен, что оставил ее где-то здесь.



Ребекка Смолли
Единственную копию моей докторской по вычислительной математике. Кажется, я ее где-то потеряла...



Эндрью Грегори
Какой-нибудь смысл... Все выглядит таким пустым, никто меня не понимает... Этот мир так жесток!



Нейл Ботвик
Душевное равновесие и номера выигрышных лотерейных билетов на следующую неделю – это неплохо, но мне хватило бы и ключей.



Дэвид Картрайт
Прочие, наверное, зададут Beagle гонку, а я предоставлю ему выходной. Пусть приносит мне тапочки.



Энди Ченнел
У меня проблема, и никто другой не сможет помочь. Если Beagle может найти ее, тогда я нанял бы The A-Team



Ричард Коббет
Источник хороших, ярких цитат, чтобы я казался остроумным, отвечая на эти вопросы.



Ричард Драммонд
Будучи слепым, как крот, я бы попросил Beagle искать мои очки, когда я их теряю.

Миссия журнала

- Пропаганда свободного ПО в России
- Продвижение решений с открытым кодом в бизнес-сообществе
- Поддержка российского Open Source сообщества
- Организация трибуны для разработчиков свободного ПО
- Обратная связь между разработчиками и потребителями ПО



Монополярность

Думаю, достаточно одного взгляда на обложку, чтобы понять – этот номер Linux Format посвящен Mono – храброй попытке перенести технологии Microsoft .NET на рельсы Open Source.

К .NET можно относиться по-разному, но, думаю, даже самый ярый скептик согласится, что эта платформа потихоньку становится стандартом де-факто для разработки ПО под Windows – не без стараний со стороны Microsoft, разумеется. И это, как ни странно, хорошо – открытая природа CLR сотоварищи означает, что новые приложения Windows без труда заработают в Linux как родные – по крайней мере с прообразом .NET, известным как Java (и ныне тоже почти открытым) все во многом обстоит именно так. Виват, Mono? А вот и нет.

Вместо того, чтобы бросить все силы на обеспечение максимального уровня совместимости, разработчики Mono идут своим путем: Gtk# является де-факто стандартом для пользовательского интерфейса Mono, а поддержка Windows.Forms все время маячит где-то на горизонте – то есть отдаляется от нас по мере приближения.

Я уверен, что у Мигеля де Икасы, равно как и у других ведущих разработчиков, имеются причины поступать именно так, а не иначе. Да, полноценная поддержка Windows.Forms во многом сродни написанию Wine 1.0. Но что получается в результате? Специализированный Linux-инструмент, который лучше Java ровно настолько, насколько Java хуже C#? **LXF**

Валентин Синицын » Главный редактор info@linuxformat.ru

Как с нами связаться

Письма для публикации: letters@linuxformat.ru
Подписка и предыдущие номера: subscribe@linuxformat.ru
Техническая поддержка: answers@linuxformat.ru
Проблемы с дисками: disks@linuxformat.ru
Общие вопросы: info@linuxformat.ru
Web-сайт: www.linuxformat.ru
› **Адрес редакции:** Россия, Санкт-Петербург, ул. Гончарная, 23, офис 54.
› **Телефон редакции:** (812) 717-00-37. Дополнительная информация на стр.128

LINUX FORMAT

Февраль 2007 № 2 (89)

Содержание

Весь номер – прямо как на ладони: приятного чтения!

Учебники



LXF DVD89

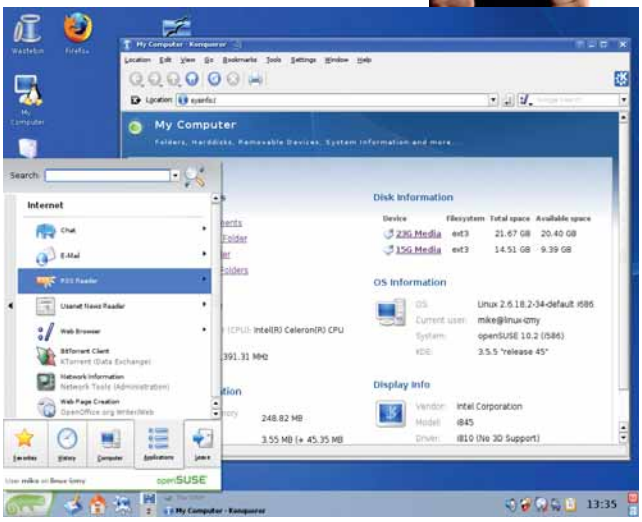


OpenSUSE 10.2
Второй по популярности дистрибутив в мире доступен для загрузки прямо с нашего DVD! Обновленные пакеты, оригинальное меню KDE и многое, многое другое...

Damn Small Linux 3.1
Нашли на антресолях завалявшийся Pentium? Водрузите на него этот легкий, не жадный до памяти дистрибутив – пусть он еще поработает.

Много Mono
После того, как вы прочитали спецрепортаж, учебник Пола Хадсона и колонку нашего редактора, грех не увидеть все своими глазами.

Журнал в PDF
Сорок две страницы из предыдущих номеров LXF: сеть, графика, *OpenOffice.org Writer* и *Impress*



Инновационное меню KDE в OpenSUSE 10.2.

Что за штука... Python 3000?

Мечта становится реальностью: узнайте самое главное с. 46





LXFHotPicks



Macromedia Flash на x86-64 – это просто!



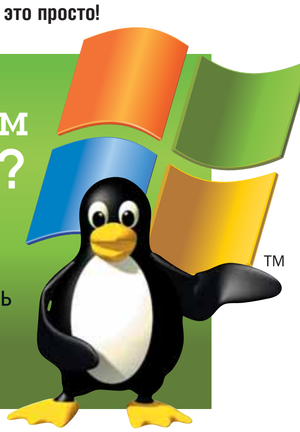
ИНТЕРВЬЮ LXF

«Я не хотел быть бизнесменом, я хотел быть программистом.»

Майкл Тиманн с. 30

Взламываем Windows? с. 38

Узнайте, как применить ваши Linux-навыки в Windows и Mac OS X!



Содержание



Подпишись на Linux Format и сэкономь!



LXF DVD внутри!

См. страницу 118

Спецрепортаж

Mono уже с нами!

Mono проложил дорогу в основные дистрибутивы Linux: почему это хорошо и как это работает?! с. 22



А также...

Постоянные рубрики



› Абсолютно открытый смартфон и много всего другого интересного – в новостях

Обзоры



› Готовы попробовать BSD?

Сравнение: трекеры



LINUX FORMAT Новости

ГЛАВНЫЕ НОВОСТИ: Разработка Neo1973 » Экономные мигрируют на Linux » Виртуальный мир Second Life » Sony Play Station 3 » Свободная виртуализация



» Рубрику ведет Илья Шпаньков



Открытый смартфон поступает в продажу

В феврале поступает в продажу первый смартфон, основанный полностью на открытых стандартах. Разработка Neo1973 началась сравнительно недавно – в ноябре 2006 года, по инициативе тайваньской компании OpenMoko (Open Mobile Kommunications), занимающейся созданием программного обеспечения и аппаратных средств с использованием свободного ПО, в содружестве с FIC (First International Computer) – известным производителем компьютеров, ноутбуков, материнских плат и аппаратной периферии. В отличие от ожидаемой к середине года основанной на Linux операционной системы для мобильных устройств от PalmSource (разработчиков Palm OS), которая будет включать в себя и проприетарные компоненты (например – интерфейс пользователя), программное обеспечение Neo1973 является полностью открытым, и, кроме того, будет сопровождаться пакетом средств разработки приложений под свободной лицензией.

Ориентировочная цена нового устройства составляет 350 долларов, и за эти деньги владелец получит стильный смартфон стандарта GSM/GPRS 850/900/1800/1900 с габаритами 120.7 x 62 x 18.5 мм, оснащенный чувствительным к касанию TFT-дисплеем стандарта VGA с диагональю 2,8 дюйма и разрешением 480x640, встроенным чипом AGPS для системы глобального позиционирования, процессором Samsung s3c2410 с частотой 266 MHz, являющимся представителем семейства SoC (System-on-a-chip – «система на чипе»), встроенными адаптерами Bluetooth 2.0 EDR и USB 1.1. Устройство оснащено двумя видами памяти, включающими 128 МБ SDRAM и 64 МБ NAND Flash, и поставляется с аккумуляторной батареей емкостью 1200 mAh, способной подзаряжаться через USB. Операционная система базируется на ядре Linux-2.6.17.14, а в качестве графической среды используются основанный на *GTK+ 2.6.10* оконный менеджер Matchbox Window Manager в связке с графическим сервером *X.org* 7.1. Предустановленное программное обеспечение обеспечивает все стандартные функции смартфона, но, по словам разработчиков, на данный момент для Neo1973 уже существует около 3000 готовых программ, управление установкой и удалением которых осуществляется с помощью специализированного Менеджера пакетов, входящего в состав базового пакета программного обеспечения.

Еще больше расширить список приложений для Neo1973 позволит выпущенный под свободной лицензией SDK для сторонних разработчиков. И такой шаг – не дань моде: компания FIC планирует использовать платформу OpenMoko в качестве базовой для всех своих смартфонов следующего поколения и большой набор прикладных программ позволит лучше конкурировать с другими участниками рынка. Таким образом Neo1973 стал первой массовой моделью нового семейства и результаты его продаж могут повлиять на дальнейшее развитие полностью открытых платформ для мобильных устройств.

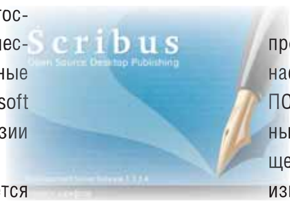
www.openmoko.com





Экономные мигрируют на Linux

Пока теоретики и экономисты ломают копья в бурных дебатах о том, достаточно ли выгодно отказываться от проприетарного ПО в пользу свободного, практики уже получают реальную прибыль от использования Free Software взамен проприетарного. Один из примеров – компания Electronics Corporation of Tamil Nadu Ltd. (ELCOT), являющаяся собственностью правительства и обеспечивающая IT-потребности организаций одного из южных штатов Индии – Тамил Наду. По словам исполнительного директора ELCOT господина Умашанкара, в большей части проектов компании используется ПО с открытыми исходными текстами, включая Linux. Это обусловлено такими факторами, как более низкая стоимость реализации подобных решений, а также простота эксплуатации и лучшая защищенность открытого ПО. «Моя обязанность – максимально экономить бюджетные средства при внедрении новых технологий и открытое программное обеспечение позволяет делать это также, если не более эффективно, чем коммерческое при максимально низких затратах» – сказал господин Умашанкар в своем интервью. В качестве примера он привел свои безуспешные попытки переговоров с корпорацией Microsoft с целью снижения стоимости одной лицензии на Windows XP Home Edition до 11 USD.



Следует отметить, что Тамил Наду является одним из штатов, играющих ключевую роль в жизни Индии, что позволяет аналитикам предсказать более активное продвижение решений на базе открытого ПО во всем государстве в целом. Еще большую реалистичность подобным прогнозам добавляют и события, происходящие в других уголках страны – в частности, в штате Керала все образовательные учреждения вольны самостоятельно выбирать, какое ПО использовать при обучении школьников и студентов, а также, по словам представителей правительства штата, власти следят за тем, чтобы при приеме на работу не было проявлений дискриминации к кандидатам в зависимости от их предпочтений в плане использования Windows или каких-либо Linux-систем.





К сожалению, господин Умашанкар не представил конкретных цифр, показывающих, насколько выгодно использование открытого ПО, но для примера можно обратиться к данным американской компании Hines Corp, осуществляющей управление несколькими производственными предприятиями на Среднем Западе и в Техасе. Их миграция с решение Microsoft на открытые программные продукты Novell, занявшая около полутора лет, позволила получить экономию в миллионы долларов. В частности, по словам руководителя информационной службы компании Эда Харпера [Ed Harper], отказ от Microsoft ERP в пользу свободных аналогов позволил снизить только начальные затраты с 1,2 млн. долларов до 400 тыс, и это только малая часть всего используемого в корпорации программного обеспечения. Харпер признался: «Представив руководству корпорации расчеты по использованию ПО от Microsoft для тех же целей, я услышал слова, которые не могу повторить прилюдно».

PS3 – инструмент разработчика

Уникальные аппаратные возможности игровой приставки Sony Play Station 3 давно вызывали интерес у людей, занимающихся созданием программного обеспечения, и сегодня они получают реальный шанс превратить развлекательное устройство в мощный инструмент разработчика. Это стало возможным благодаря новой инициативе компании Terra Soft в содружестве с RapidMind, совместными усилиями создавшими пакет инструментальных средств программиста RapidMind Development Platform v2.0, позволяющих наиболее полно использовать аппаратные возможности PS3 при создании приложений. В качестве «фундамента» данная платформа использует операционную систему Yellow Dog Linux 5.0 от компании TerraSoft, обеспечившей полноценное функционирование «взрослой» ОС на игровой приставке, а RapidMind Development Platform позволяет освоить все возможности технологии Cell Broadband Engine™ (Cell BE), используемой в PS3.

Ранее разработчикам приходилось подробно изучать архитектуру игровой приставки, представляющей из себя своего рода мини-суперкомпьютер, состоящий из восьми процессоров. В связи с этим появление новых приложений, способных функционировать на ней, было связано с большими трудностями. Теперь же, благодаря новой платформе разработки от RapidMind, автор программы может не задумываться о том, как максимально полно использовать возможности многопроцессорной архитектуры – данную обязанность берет на себя RapidMind Development Platform. Таким образом, появление этого программного продукта еще больше приближает людей, желающих превратить PS3 в полноценный мощный компьютер, к реализации своих планов.

rapidmind.net



Новости короткой строкой

» Компания ASPLinux выпустила DVD диск с обновлением операционной системы ASPLinux 11.2.

» По результатам третьего квартала 2006 года, инициатива Oracle Unbreakable Linux не повлияла на дальнейший рост прибыли компании Red Hat.

» Издание InfoWorld Technology присвоило дистрибутиву SUSE Linux Enterprise Desktop 10 звание «лучшая настольная операционная система».

» Начиная с версии 7, дистрибутив Fedora не будет разделяться на Core и Extras, а будет распространяться единым набором приложений.

» Новая версия интернет-планшета Nokia N800 Internet Tablet по непонятным причинам появилась в американских магазинах раньше официального анонса.



» Юристы компании Novell считают, что SCO Group обанкротится в любом случае, а иск Novell не способствует этому процессу, а является лишь попыткой вернуть себе незаконно присвоенные ответчиком средства до этого момента.

«Вторая жизнь» доступна всем. Частично…

Один из самых необычных онлайновых ресурсов – виртуальный трехмерный мир «Second Life», приобщился к миру Free Software благодаря недавним революционным решениям. В частности, компания Linden Research, Inc., являющаяся автором этого проекта, опубликовала исходные коды клиентской части сложного программного комплекса под свободной лицензией. Таким образом пользователи Windows, Linux и Mac OSX получили в свое полноправное пользование уникальный инструмент, который позволит сделать виртуальный мир Second Life еще более разнообразным и фантастичным.

Следует напомнить, что Second Life – это не просто онлайновая игра, а нечто большее. Пользователи данного ресурса с помощью оригинального скриптового языка могут создавать трехмерные объекты, включающие не только собственно аватары участников, но и неодушевленные предметы – различные сооружения, элементы ландшафта и даже целые острова. Внутри этого необычного мира обращается собственная валюта и любой из участников может организовать собственный бизнес-проект и даже зарабатывать реальные деньги, переводя «Линдены» по особому курсу на личный банковский счет. По сути, Second Life полностью оправдывает свое название и позволяет любому желающему попытаться вновь реализовать свои идеи или проекты, но уже на новом, виртуальном уровне. Такой шанс оказался востребован сотнями тысяч людей и в короткие сроки (проект стартовал в 2003 году) Second Life стал настолько популярен, что даже крупные компании пожелали вложить деньги и открыть свои представительства в нереальном мире.

К слову, открытие исходных текстов обсуждалось в недрах компании-разработчика около трех лет. В течении этого времени многие компании, открывшие свои представительства в мире Second Life, получили доступ к исходным кодам программы в целях наилучшего функционирования собственных виртуальных филиалов. В добавок к этому сами пользователи ресурса проявили недюжинную активность – по данным компании Linden, около 15% участников на сегодняшний день генерируют около 7 миллионов строк кода ежедневно с помощью скриптового языка, являющегося встроенным инструментом сервиса, в результате чего общий объем кода Second Life уже достиг 5 ГБ. По задумке разработчиков, открытие исходного кода позволит поднять качество программного обеспечения виртуального мира на новый, ранее недоступный уровень.

http://secondlife.com/developers/opensource/



Свободная виртуализация





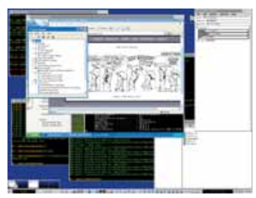












Системы виртуализации, позволяющие одновременно осуществлять на компьютере работу в нескольких абсолютно разных операционных системах, в последнее время стали одним из наиболее приоритетных направлений развития ИТ-индустрии. В мире Free Software существует несколько вариантов подобного ПО, начиная от коммерческого VMWare и заканчивая включением в ядро Linux соответствующих компонентов, однако полного виртуализатора (то есть решения, не требующего модификации гостевой ОС или аппаратной поддержки технологий виртуализации хост-процессором) до сих пор не существовало. В середине января данный недостаток был устранен – разрабатываемый германской компанией InnoTek пакет VirtualBox опубликован под свободной лицензией (GPL). Отныне все желающие могут не только использовать VirtualBox в различных целях, но и принять участие в дальнейшем совершенствовании этого, безусловно, полезного программного продукта.

В качестве базовой операционной системы для установки VirtualBox могут использоваться Windows или Linux, при этом обеспечивается полноценное выполнение множества «гостевых» систем, начиная от DOS и заканчивая OpenBSD. Данный пакет доступен в нескольких вариантах: VirtualBox Open Source Edition (OSE) распространяется в виде исходных кодов под свободной лицензией; закрытый VirtualBox, отличающийся от VirtualBox OSE поддержкой USB, iSCSI и удаленного доступа к виртуальной машине, доступен бесплатно для личных и ознакомительных целей. Создатели VirtualBox не исключают, что со временем некоторые функции перекочуют из закрытой редакции VirtualBox в открытую.

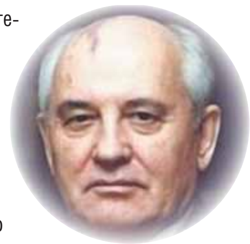
http://www.virtualbox.org/

Особенности национальной информатики



Так называемое «Дело По́носова», всколыхнувшее российское околокомпьютерное сообщество и породившее массу обсуждений как на уровне форумов, так и в высших кругах власти, на самом деле поражает своей нелепостью. Между тем, несмотря на подобную оценку, поддерживаемую большинством наблюдателей, в ходе разбирательств обнажились проблемы, которые не один год являются актуальными для российских пользователей. Напомним, что директор сельской школы, расположившейся в поселке Сепычев (Пермская область), 40-летний Александр Михайлович По́носов (на фото) оказался под следствием в связи с тем, что на 12 компьютерах, купленных школой для оборудования компьютерных классов, обнаружилось контрафактное ПО производства корпорации Microsoft. По подсчетам следственных органов, общий ущерб американского правообладателя составил более 266 тысяч рублей, в связи с чем данное «деяние» грозит директору школы лишением свободы на срок до 5 лет. Между тем, сам обвиняемый не признает своей вины – компьютеры были поставлены уже с предустановленным ПО, поэтому он не имеет никакого отношения к нарушению прав разработчика – корпорации Microsoft.

Следует отметить, что данное судебное дело инициировано не американской компанией, которая, кстати, лично директору школы никаких претензий не предъявляла, а российскими правоохранительными органами, следящими за тем, чтобы отечественные пользователи, как персональные, так и корпоративные, не работали на контрафактном ПО. Также выяснилось, что основной виновник – работник частного предприятия, готовивший компьютеры к передаче учебному заведению и установивший те самые нелицензионные копии операционной системы и пакета офисных приложений, уже понес наказание от своего руководства в виде штрафа размером 10000 рублей. Таким образом, собственно директор школы оказался, скорее, стороной потерпевшей, чем виновником нарушения закона. Эту точку зрения разделяют и многие его односельчане – одно из судебных заседаний по данному делу сопровождалось пикетом в поддержку директора, организованным учениками его школы. Более того – на стороне По́носова выступили и известные политические деятели, включая бывшего президента СССР Михаила Горбачева, написавшего открытое письмо владельцу Microsoft Биллу Гейтсу, и действующего президента России Владимира Путина, назвавшего в одном из интервью ситуацию с делом По́носова «собачьей чушью». По словам же директора Роспечати Михаила Сеславинского, «если начать по всей строгости закона наказывать пользователей, то в тюрьме может оказаться половина населения страны, включая и самих работников правоохранительных органов».



Можно с достаточной долей уверенности предположить, что судебные разбирательства с участием директора сельской школы в качестве обвиняемого в конце концов закончатся «без особых последствий» для него лично, но это совсем не значит, что инцидент пройдет незаметно для нашего общества в целом. Руководство Пермского края, а вслед за ним и столичные государственные организации всерьез задумались о том, какие подводные камни могут оказаться на пути реализации многих важных для страны программ при выборе в пользу коммерческого ПО. В связи с этим можно прогнозировать повышение интереса к свободному программному обеспечению, в большинстве случаев являющемуся вполне полноценной альтернативой коммерческим продуктам. Также есть надежда, что подобные последствия «Дела По́носова» станут еще одним шагом к искоренению, пожалуй, наиболее важной проблемы, поднятой в результате данного инцидента – слишком несерьезному отношению российских компаний и частных лиц к использованию пиратского программного обеспечения. »





› **«Билл, ты гонишь!»**
Главу Microsoft в свое время тоже арестовывали – за превышение скорости...

ОТ РЕДАКЦИИ

После того, как «дело По́носова» получило широкую огласку, на наши ящики стали приходить в буквальном смысле сотни писем примерно следующего содержания: «У нас в организации имеется нелицензионное ПО. Что делать?». Не претендуя на истину в последней инстанции, предлагаем следующую программу действий:

Первым делом – приобрести OpenOffice.org 2.1, лучше всего в сборке «Инфра-Ресурса». Ее также можно загрузить свободно, но в покупной версии есть бумажная лицензия, которую можно продемонстрировать в случае проверки. Весной 2007 года выйдет книга по OpenOffice.org 2.1, изданная LinuxCenter.Ru и BHV – в ней также будет содержаться текст лицензионного соглашения и компакт-диск с OOo для Linux и для Windows. Просто заменив пиратский Microsoft Office на OpenOffice.org, вы снизите общую сумму ущерба, нанесенного правообладателю, более чем вдвое.

Второй (по сложности реализации и первый – по лицензионной чистоте) вариант – полная миграция на Linux. Дистрибутивы можно бесплатно загрузить из Сети (подборку наиболее популярных вариантов можно также найти на ftp.linuxcenter.ru), взять с прилагаемых к журналу дисков или приобрести коробочную версию – например, Mandriva Linux 2007, MOPS Linux или Scientific Linux. В комплекте с коробочным дистрибутивом идет текст лицензионного соглашения, которое подтверждает правомочность использования системы. Следует отметить, что коробочные версии в большинстве случаев допускают установку на произвольное число рабочих мест (это может не относиться к входящему в коробочный дистрибутив коммерческому ПО), при этом поддержкой от производителя будет пользоваться только один из них. Таким образом, вы можете выбирать, сколько коробок необходимо закупить для нужд школы или другой организации.

Для быстрого обучения Linux-технологиям рекомендуем хорошо зарекомендовавший себя курс «Основы работы в Linux», доступный по адресу http://www.intuit.ru/department/os/baselinuxwork/. Несмотря на то, что дистрибутивы, на основе которых рассматриваются учебные вопросы, не самых последних версий, курс содержит все сведения, которые необходимы для уверенной работы в среде Linux. В скором времени (возможно, к тому моменту, когда вы будете читать эти строки) в дополнение к «Основам работы в Linux» будет выпущен еще один курс по OpenOffice.org.

Конкурс на лучшую статью о пакете OpenOffice.org

Неформальное сообщество разработчиков и пользователей **community.i-rs.ru** при поддержке журнала Linux Format и компании «Инфра-Ресурс» объявляет конкурс среди любителей и профессионалов на лучшую статью о пакете *OpenOffice.org*.

Генеральный спонсор конкурса – журнал Linux Format.

Призовой фонд конкурса – 50000 рублей.

Победителям – годовая подписка на журнал Linux Format.

Всем участникам конкурса компания «Инфра-Ресурс» вручает значки *OpenOffice.org*.

Цели Конкурса

•Выявить лучших технических писателей среди профессионалов и любителей.

•Добиться, чтобы статей о пакете *OpenOffice.org* на русском языке стало больше, а их авторы упоминались чаще.

Задачи Конкурса

•Популяризация пакета *OpenOffice.org*

•Распространение передового опыта использования OpenOffice.org

•Создание открытого тематического ресурса в сети Интернет, посвященного вопросам эффективной эксплуатации *OpenOffice.org*.

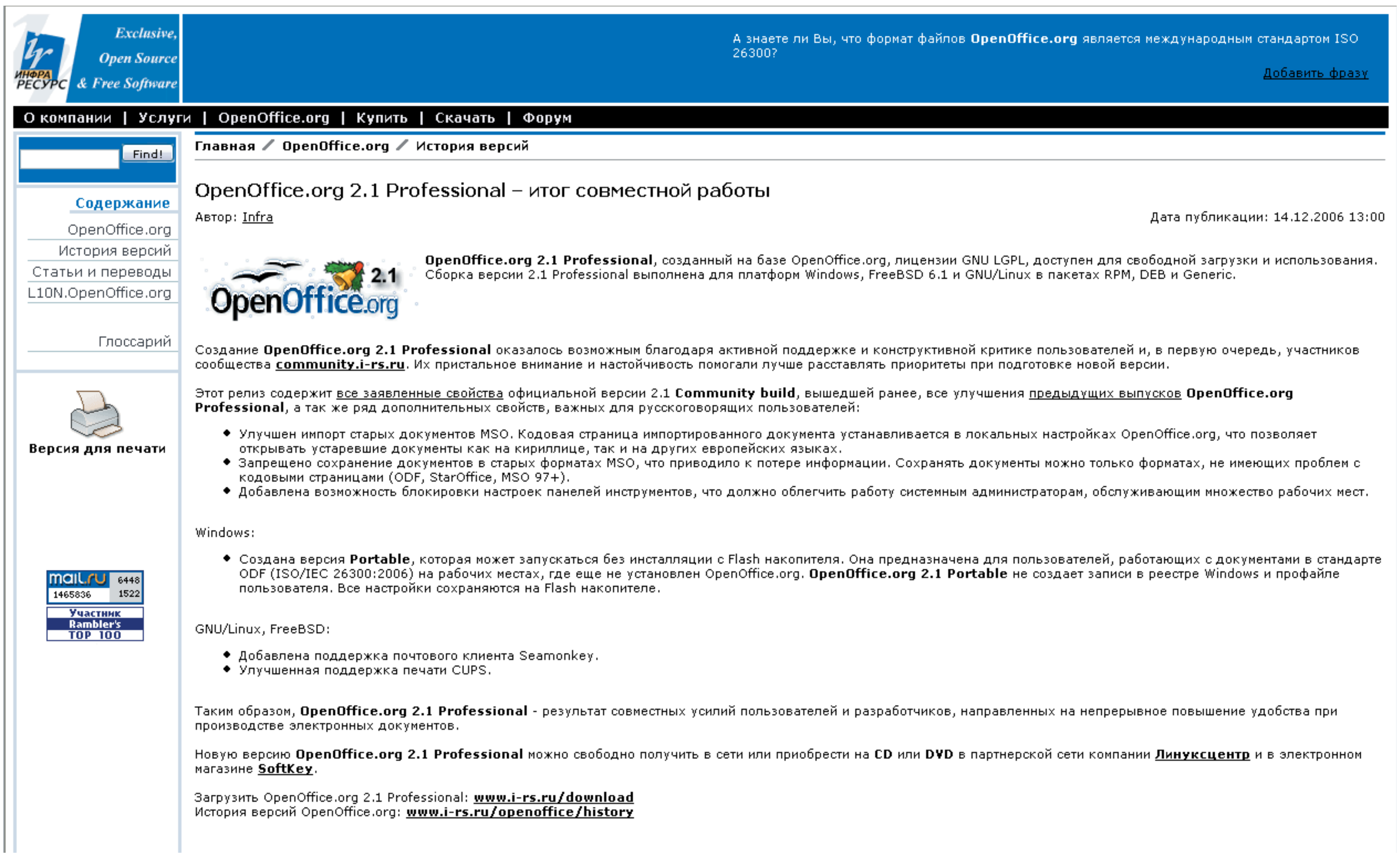
Конкурс проводится в пяти номинациях:

1. **«ОФИСНЫЕ ТЕХНОЛОГИИ»** – статьи о новых и существующих решениях для производства электронных документов в формате ODF (OpenDocument Format – ISO/IEC 26300:2006), приёмах эффективной эксплуатации *OpenOffice.org*;
2. **«ТЕСТЫ, СРАВНЕНИЯ, ОБЗОРЫ»** – аналитические статьи и материалы о пакете *OpenOffice.org*, достоинства и недостатки;
3. **«УЧИМ РАБОТАТЬ С OPENOFFICE.ORG»** – методические пособия и учебные курсы по работе с *OpenOffice.org*;
4. **«ИСТОРИЯ ОДНОГО ПРОЕКТА»** – «истории успеха», правдивые и весёлые истории из практики внедрения и эксплуатации *OpenOffice.org*;
5. **«ОБ ЭТОМ УЖЕ ПИСАЛИ»** – переводы статей зарубежных авторов об *OpenOffice.org*.

Приглашаем авторов и спонсоров.

Регламент конкурса опубликован по адресу: **http://www.linuxformat.ru/contest/ooo2006.phtml**

Подробнее о конкурсе можно узнать в сети по адресу: **http://www.i-rs.ru/openoffice/contest**



Exclusive, Open Source & Free Software

А знаете ли Вы, что формат файлов **OpenOffice.org** является международным стандартом ISO 26300?

Добавить фразу

О компании | Услуги | OpenOffice.org | Купить | Скачать | Форум

Find!

Содержание
OpenOffice.org
История версий
Статьи и переводы
L10N.OpenOffice.org
Глоссарий

Версия для печати

Главная / OpenOffice.org / История версий

OpenOffice.org 2.1 Professional – итог совместной работы

Автор: Infra

Дата публикации: 14.12.2006 13:00

OpenOffice.org 2.1 Professional, созданный на базе OpenOffice.org, лицензии GNU LGPL, доступен для свободной загрузки и использования. Сборка версии 2.1 Professional выполнена для платформ Windows, FreeBSD 6.1 и GNU/Linux в пакетах RPM, DEB и Generic.

Создание **OpenOffice.org 2.1 Professional** оказалось возможным благодаря активной поддержке и конструктивной критике пользователей и, в первую очередь, участников сообщества **community.i-rs.ru**. Их пристальное внимание и настойчивость помогали лучше расставлять приоритеты при подготовке новой версии.

Этот релиз содержит все заявленные свойства официальной версии 2.1 **Community build**, вышедшей ранее, все улучшения предыдущих выпусков **OpenOffice.org Professional**, а так же ряд дополнительных свойств, важных для русскоговорящих пользователей:

- Улучшен импорт старых документов MSO. Кодовая страница импортированного документа устанавливается в локальных настройках OpenOffice.org, что позволяет открывать устаревшие документы как на кириллице, так и на других европейских языках.
- Запрещено сохранение документов в старых форматах MSO, что приводило к потере информации. Сохранять документы можно только форматах, не имеющих проблем с кодовыми страницами (ODF, StarOffice, MSO 97+).
- Добавлена возможность блокировки настроек панелей инструментов, что должно облегчить работу системным администраторам, обслуживающим множество рабочих мест.

Windows:

- Создана версия **Portable**, которая может запускаться без инсталляции с Flash накопителя. Она предназначена для пользователей, работающих с документами в стандарте ODF (ISO/IEC 26300:2006) на рабочих местах, где еще не установлен OpenOffice.org. **OpenOffice.org 2.1 Portable** не создает записи в реестре Windows и профайле пользователя. Все настройки сохраняются на Flash накопителе.

GNU/Linux, FreeBSD:

- Добавлена поддержка почтового клиента Seamonkey.
- Улучшенная поддержка печати CUPS.

Таким образом, **OpenOffice.org 2.1 Professional** - результат совместных усилий пользователей и разработчиков, направленных на непрерывное повышение удобства при производстве электронных документов.

Новую версию **OpenOffice.org 2.1 Professional** можно свободно получить в сети или приобрести на **CD** или **DVD** в партнерской сети компании **Линуксцентр** и в электронном магазине **SoftKey**.

Загрузить OpenOffice.org 2.1 Professional: **www.i-rs.ru/download**
История версий OpenOffice.org: **www.i-rs.ru/openoffice/history**

LINUX FORMAT

Обзоры



Новинки программного и аппаратного обеспечения в описании наших экспертов



Алексей Федорчук
Свою первую (и последнюю) программу написал еще на Алголе.

Debian или Kebian?

Семимильными шагами приближается день релиза очередного Debian, известного под партийной кличкой Etch. Так что перед нами последний шанс ознакомиться с тем, что будет – до того, как это будущее настанет.

Как? Самый простой способ – заглянуть на страницу, с которой можно скачать официальные снимки тестируемой версии, обновляемые еженедельно (http://cdimage.debian.org/cdimage/weekly-builds/). Здесь мы увидим полный слепок дистрибутива в его текущем состоянии. Ныне он насчитывает 22 диска, пронумерованных, как ни странно, с 1-го по 22-й. Но что мы видим в конце? Еще два образа первых дисков – **debian-testing-i386-kde-CD-1.iso** и **debian-testing-i386-xfce-CD-1.iso**. С помощью дедуктивного метода товарища Ш.Холмса не трудно догадаться, что второй из первых дисков предназначен для установки Debian с KDE в качестве рабочего стола по умолчанию, третий же предлагает в этом качестве среду XFce. Что же лежит на «первом» первом диске? Элементарно, Ватсон – методом исключения приходим к выводу, что на нем будет не иначе как GNOME.

Теперь остается только скачать какой-либо образ и проверить свои подозрения. Я, разумеется, проделал это с диском, подозрительным на присутствие KDE. И что же оказалось после установки с него? Оказалось, что, если инсталлировать Debian методом цыпленка, клюющего клавишу Enter, мы безальтернативно, даже в режиме эксперта, получаем рабочую станцию с KDE in corpore – включая kdeedu, kdegames, kdetoys. Благо, хоть без всех мыслимых и немыслимых локалей, входящих в состав kde-i18n. Будет в нашем распоряжении и kdewebdev – а вот собственно средств разработки KDE не окажется. И, как ни странно, не найдем мы в инсталлированной системе и *KOffice* – место его займет «универсальный» *OOo*.

По аналогии можно сделать умозаключение, что при умолчальной инсталляции с «первого» первого диска мы получим рабочую станцию GNOME, а с диска третьего – ее же, но в XFce-обрамлении. Ничего не напоминает? Если вы скажете, что напоминает Ubuntu, Kubuntu и Xubuntu – не смогу возразить. Так что же, теперь у нас вместо Debian'а будут Ге-биан, Ке-биан и Хе-биан? Можно было бы сказать и так. Однако установку с диска netinstall пока не отменили – и посредством него можно обзавестись базовой системой, которую останется только нарастить по собственному усмотрению. В общем, можно только повторить слова нашего великого Генсека: «Мне нравится».

alv@posix.ru

Сегодня мы рассматриваем...



CrossOver с. 10



› **Здорово заполучить кучу игр под Linux, но как их там запустить? Тестируем *CrossOver Linux*...**

Ardour с. 12



› **Да, интерфейс – не подарок, но освоив его, вы станете круче Жан-Мишеля Жарра!**

НАШ ВЕРДИКТ: пояснение

Все попавшие в обзор продукты оцениваются по одиннадцатибальной шкале (10 – высшая оценка, 0 – низшая). Как правило, мы оцениваем функциональность, производительность, простоту использования и цену, а для бесплатных программ учитывается документация. Кроме того, мы всегда выставляем общую оценку, демонстрирующую наше отношение к продукту.

Выдающиеся решения могут получить престижную награду «Top Stuff». Номинантами становятся лучшие из лучших – просто высокой оценки здесь недостаточно.

LINUX FORMAT TOP STUFF

Рассматривая свободное ПО, мы обычно указываем предпочтительный дистрибутив. Иногда это означает компиляцию из исходных текстов, но, если разработчики рекомендуют Autopackage, мы следуем этому совету.

LINUX FORMAT Вердикт

Google Earth

Разработчик: Google
Сайт: http://earth.google.com
Цена: Бесплатно по закрытой лицензии

Функциональность	10/10
Производительность	9/10
Простота использования	9/10
Документация	9/10

» *Если весь мир – сцена, то Google Earth – театр. Простая в использовании, захватывающая и ободряюще практичная программа.*

Рейтинг 9/10

CrossOver Linux 6.0

Угрожает ли Windows коммерческий вариант *Wine*? Двухсистемный **Алек Мир** взвешивает шансы.

Вкратце...

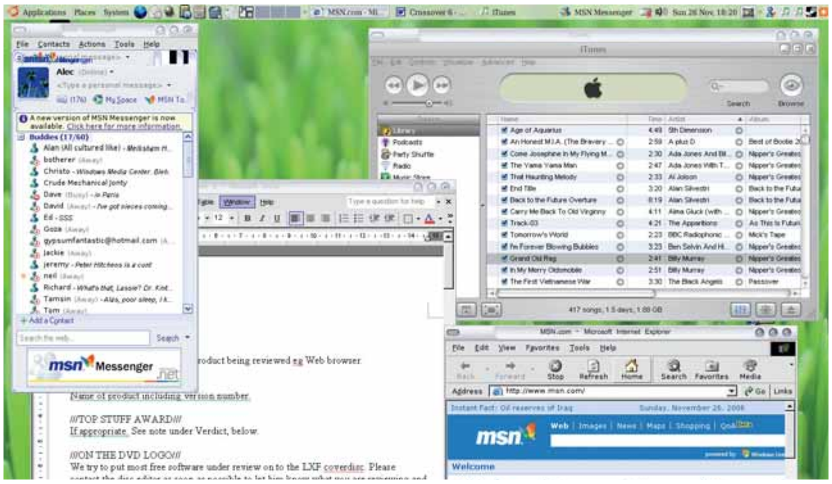
» Эмулирует рабочую среду Windows для отдельных приложений. См. также: *Cedega*.

Иногда вам надоедает *Frozen Bubble* и возникает желание пострелять… вдоволь пострелять. *CrossOver* от CodeWeavers утоляет жажду насилия. С помощью коммерческой, изрядно модифицированной версии *Wine* становится возможным (теоретически) запускать некоторые «насущные» программы – *Microsoft Office*, *Adobe Photoshop*, *Dreamweaver*, *Counter-Strike* и *World of Warcraft* – внутри вашего дистрибутива, и позволит обойтись без Windows CD на вашем экологически чистом Linux-компьютере.

Главное, за что вы платите в *CrossOver* – это графический интерфейс, выдающееся свойство программы. Инсталлируется *CrossOver* всего одной командой, и по умолчанию (на Ubuntu Dapper) вьет себе гнездо в домашней директории. После этого вы попадаете прямо в графический интерфейс. При его сложности это замечательно тонкая штука, но некоторой важной информации явно недостает. Например, большинство программ жалуются на отсутствие *Internet Explorer* – было бы неплохо предупреждать, что надо установить его, а также некоторые другие зависимости, перед работой.

Если забыть об этой нестыковке, инсталляция приложений необычайно проста, ноль проблем. Любая программа изящно интегрируется в рабочую среду, и приложения Windows запускаются от своих ярлыков без намека на программу-посредника – как естественная часть рабочего стола. Например, значок *MSN Messenger* спокойно занимает место в системном лотке, а новые приложения расценивают вашу файловую систему как Windows-папки или диски, несмотря на то, что в ней ничего не напоминает FAT32 или NTFS. Чрезвычайно интеллигентный софт.

Увы, мы натолкнулись на многократные ошибки отображения и неработающие органы управления. Например, установить *Microsoft Office XP* удалось только с восьмой попытки. Чтобы добиться этого, нам пришлось заменить подходящую по всем параметрам «бутылку» (изолированную эмулируемую среду) Win98 на Win XP, которая была помечена как неподдерживаемая. Справедливости ради, стоит отметить, что я все же пишу эти слова в Office XP, запущенном из-под Ubuntu. Неподдерживаемые приложения были еще менее успешны – ошибки варьировались от простого бездействия до пугающей невозможности нажать кнопку Next во время инсталляции. Как тут не обозлиться, за свои-то денежки (правда, гарантия позволяет их вернуть), но обычные покупатели получают шестимесячную поддержку и бесплатное обновление. Плюс сознание, что они вложились в проект *Wine*.



› **Полноценный рабочий стол как у Microsoft, только в Ubuntu. Круто, правда?**

Выпустить пар

Игры оставили смешанное чувство. Конечно, есть теоретическая возможность запускать неподдерживаемое ПО, но с более чем полудюжиной новейших игр (включая *Quake 4*) мы имели нулевой успех. Заставить работать удалось лишь игры, прямо указанные в списке *CrossOver*: *World of Warcraft* и *Half-Life 2/Counter-Strike* (с помощью онлайн-приложения Steam [Пар]). И хотя игры для *CrossOver* идут всего лишь как бонус, это два известнейших компьютерных брэнда и главное оружие в борьбе за признание Linux в качестве игровой платформы. Производительность первой из них оказалась ничтожной: играть она играла, но подняться выше скорости 25 кадров в секунду не удалось (на Windows – 46), а экран покрывался квадратиками при смене разрешения с 1024x768. Намного лучше дела обстоят с *Half-Life 2* и *Counter-Strike*, они работают гладко (хоть и намного медленнее, чем на Windows), на 50 кадрах. Наша DirectX 9 видеокарта Nvidia GeForce 7800 была опознана как поддерживающая только DirectX 8, поэтому игры шли при заметно сниженном визуальном качестве. Нечего и говорить, что пакет дополнений для Half-Life 2, Episode 1, не смог запуститься.

И все же впечатления остались приятные, а известие о включении в следующую версию *CrossOver* эмуляции DX9 еще более подогревает интерес к программе. LXF



Свойства навскидку



World of Warcrash
Производительность *WoW* оставляет желать лучшего – но она работает. Мило с ее стороны.



Полу-Half-Life
Half-Life 2 работает терпимо, хотя и с чудовищно заниженной детализировкой.

LINUX FORMAT Вердикт

CrossOver Linux 6.0

Разработчик: CodeWeavers
Сайт: www.codeweavers.com
Цена: $39.95

Функциональность	7/10
Производительность	8/10
Простота использования	7/10
Цена	8/10

» *Отлично работает с Office, IE и Photoshop, но нуждается в расширении зоны охвата и в отладке.*

Рейтинг 8/10

Damn Small Linux 3.1

Бывают ли Live CD еще меньше? Много ли толку в 50 МБ? **Нейл Ботвик** ищет ответы на большие вопросы маленького дистрибутива.

Вкратце...

» 50 Мб Live CD, может работать с USB-устройства. См. также: Puppy Linux.

Если быть точным, Damn Small Linux – это вариант Knoppix, ужатый до 50 Мб: размера компакт-диска – «визитки» или 1,5% от объема DVD Fedora Core 6. Но эти 50 Мб содержат невероятное количество приложений: браузеры *Dillo* и *Firefox*, почтовый клиент *Sylpheed*, несколько текстовых редакторов (какой же Linux без соперничества редакторов?), службу мгновенных сообщений, графические просмотрщики и редакторы, медиа-плейеры, инструменты удаленного рабочего стола, FTP и web-серверы. Даже парочка игрушек имеется.

Для экономии места приложения лишены документации (помощь найдется на сайте DSL, форумах и wiki), да и версии не самые последние (*Firefox 1.0.6*, а ядро серии 2.4). Однако многие новшества из серии 2.6 были перенесены в 2.4, и теперь эти ядра могут работать на новейшем оборудовании. При этом ядра 2.4 лучше совместимы со старыми компьютерами, на которых обычно и крутится DSL.

Дисками-«визитками» пользуются немногие, а USB-брелки имеют намного большие объемы, но предел 50 МБ остается непреложным. FAQ поясняет, что превышение лимита сделает DSL «просто дистрибутивом»: BIOS на некоторых старых машинах не умеют загружаться с USB-носителей или с разделов свыше 256 Мб. Это значит, что дистрибутив просто не загрузится с диска объемом 700 Мб – и это прежде, чем вы сможете сохранить свой первый файл.

Если вам нужно дополнительное ПО, DSL можно переработать (о том, как это сделать, см. LXF74/75), но проще воспользоваться MyDSL, способом добавки приложений во время загрузки дистрибутива. К загрузке готово множество пакетов, а можно собрать собственные, следуя инструкциям DSL-wiki. Инсталляция пакетов сводится к копированию их на загрузочный носитель или помещению на жесткий диск с последующей передачей параметра **mydsl=hda1** во время загрузки. Можно предусмотреть автоматическое сохранение вашей домашней директории при выключении – на винчестере или на USB-устройстве.

Дистрибутив скромен и в своих системных требованиях. Мы смогли запустить X на системе с 24 Мб памяти без раздела подкачки, хотя на 32 Мб работать комфортнее. Стандартный оконный менеджер для DSL – *Fluxbox*, умелый и быстрый, хотя и требует некоторой притирки, если вы привыкли к KDE или Gnome.



› И все эти программы, плюс инструменты настройки, на ОС размером 50 МБ!



Майк считает...

«Слишком много старых компьютеров были заброшены как бесполезные. Damn Small Linux – рождественская сказка для компьютерного мира, а 3.1 – еще одна достойная версия»

Инсталляция на USB или HD

DSL содержит скрипты для установки на винчестер или USB-носитель, а также для изменения образа CD, хотя с применением MyDSL необходимость в последнем отпадает. Инсталляция на жесткий диск возможна в двух вариантах. Standard дает вам просто маленький дистрибутив Debian. Толку в нем немного – если нужен малый Debian, проще установить минимальный набор пакетов с Debian CD. Вариант Frugal (Бережливый) куда интереснее. Он создает спасательный раздел на винчестере, потребляющий всего 50 МБ, плюс место для необходимых дополнений, пользовательских настроек и пакетов MyDSL. Если вы имеете привычку губить свою систему, а затем восстанавливать ее с помощью Live CD, добавление такого раздела в загрузочную запись даст вам запасной вариант, чтобы не рыться всякий раз в колышущейся стопке дисков.

Главный соперник DSL – Puppy Linux, чуть больший по размерам (70 МБ) – оставляет DSL легчайшим из известных Live CD. Оба грузятся очень быстро (намного меньше минуты на ноутбуке с Celeron пятилетней давности), но DSL использует систему определения оборудования от Knoppix, поэтому умеет самонастраиваться на большинстве компьютеров. У DSL больше возможностей для расширения, тогда как Puppy содержит больше приложений.

При малом размере, DSL обладает внушительным набором функций, но по сравнению с большими дистрибутивами, конечно, ограничен. Так нужен ли он вообще? Несомненно, да: как облегченный вариант для старых ПК, где ядро 2.4 может стать решающим фактором (Puppy работает на 2.6.18.1); как портативная система на USB-брелке; или как страховочный раздел, притаившийся в уголке винчестера. DSL умеет намного больше, чем можно предположить, глядя на его малые размеры. LXF

> «Возможен запуск X на системе с 24 Мб RAM.»

Linux Format Вердикт

Damn Small Linux 3.1

Разработчик: Джон Эндрюс, Роберт Шинглдекер и др.
Сайт: www.damnsmalllinux.org
Цена: Бесплатно под GPL

Функциональность	8/10
Производительность	10/10
Простота использования	7/10
Документация	8/10

» *Функционально ограничен, если не обращаться к модулям MyDSL; зато легок и скор, особенно на старых машинах.*

Рейтинг 8/10

Ardour 2.0

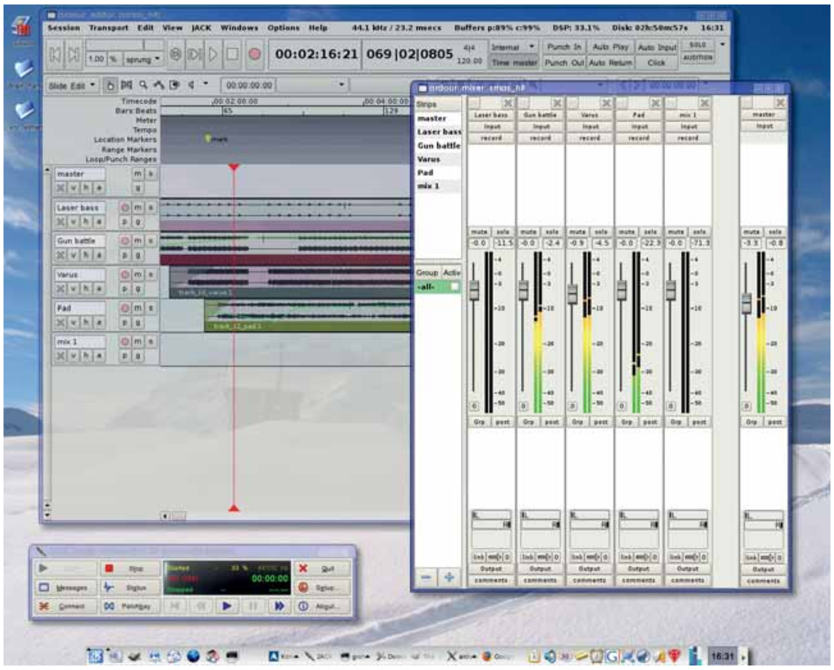
Ardour 1.0 был первым обнаруженным нами звукозаписывающим инструментом студийного уровня под Linux. **Грэм Моррисон** следит за развитием проекта.

Вкратце...

» Инструмент многодорожечной звукозаписи с эффектами реального времени, микшированием и автоматизацией. Здесь нет MIDI-секвенсора, но гибкость возможностей позволяет обойтись без него. См. также: *Muse* или *Rosegarden*.

Человек несведущий примет *Ardour* скорее за пульт управления электростанцией, чем за инструмент звукозаписи – только гляньте на снимок экрана. Не припомним другого Linux-приложения с таким количеством ползунков, кнопочек и мигающих огоньков.

Ardour – виртуальный эквивалент подключения многодорожечного магнитофона к микширующей консоли. Именно подключение придает программе замечательную гибкость. *Ardour* во всех соединениях на 100% зависит от Jack Audio Connection Kit, и даже стереоприветствие, встречающее каждый новый проект, попадает на выход вашей аудиокарты через его виртуальные кабели. Объединение консоли с магнитофоном доведено здесь до совершенства, и хотя профессионалу это дает неограниченную власть над процессом звукозаписи, для новичка все выглядит весьма и весьма пугающе.



» **Интерфейс пользователя озадачивает, но *Ardour* лишь эмулирует процесс, уже полвека протекающий в большинстве студий звукозаписи.**

Здорово – и непросто

Но время идет – и с тех пор, как мы рассматривали *Ardour 1.0* в LXF65, многое изменилось. Для начала скажем, что интерфейс перекочевал с закосневшего *GTK 1.2* на прекрасный *GTK 2.0*. Попутно разработчики внесли несколько очень полезных усовершенствований. *Ardour* получил передвижную панель инструментов для основных функций, раскладка основного окна также была улучшена. В меню теперь полно всяких полезных опций – даже, пожалуй, слишком: только в меню Transport [Перемещение] имеется 24 позиции (а мы-то привыкли видеть лишь «Играть», «Пауза», «Стоп», «Перемотка» да «Быстрая перемотка»!). Переделан неуклюжий и устаревший аудиоимпорт, хотя импортировать файлы OGG напрямую почему-то еще нельзя. Хотелось бы также видеть значки для основных органов управления: большинство из них по-прежнему обозначается цифрами и буквами (M для Mute или S для Solo). Но, в общем и целом, различные усовершенствования делают *Ardour 2.0* значительно удобнее в работе.

Еще одно дополнение версии 2.0 – удаленный контроль над внешними устройствами. Это значит, что можно управлять некоторыми кнопками и ползунками внутри *Ardour* с внешнего MIDI-устройства, что еще более стирает грань между виртуальным и реальным оборудованием. Более того, *Ardour* может посылать контрольные сигналы обратно на удаленное устройство, и состояние регуляторов и индикаторов будет точно таким, какое задано на мониторе. Объединение удаленного контроля с автоматизацией параметров позволяет построить вокруг *Ardour* солидную музыкальную студию. Еще одно свойство, ценное для профессионала: история откатов теперь хранится вместе с проектом, и любые действия предыдущей сессии можно отменить.

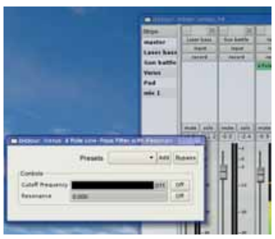
Несмотря на все эти новинки, некоторые базовые функции по-прежнему ждут своего часа. Работа с аудиоэффектами нуждается в усовершенствовании. В настоящее время они просто идут один за другим, без всякой попытки классификации, а ведь вариантов сотни. Необходимо иерархическое разделение эффектов по типу на динамические, временные и модуляционные.

А главная беда *Ardour* – прямое следствие его совершенства: фантастическая гибкость достигается ценой невероятной сложности, и неспециалисту он не по плечу. Но для опытных аудиоинженеров *Ardour* может стать реальным соперником коммерческих приложений других платформ. LXF

Свойства навскидку



Контроль громкости
Неразрушающее микширование означает, что можно управлять громкостью в реальном времени.



Регуляторы эффектов
Эффекты реального времени имеют собственные ползунки, которые в свою очередь можно автоматизировать.

LINUX FORMAT Вердикт

Ardour 2.0

Разработчик: Пол Дэвис [Paul Davis] и др.
Сайт: http://ardour.org
Цена: Бесплатно под GPL

Функциональность	9/10
Производительность	8/10
Простота использования	4/10
Документация	6/10

» *Его подкосил низкий балл за простоту использования, но Ardour дьявольски изощрен, и заслуживает серьезного внимания.*

Рейтинг 7/10

OpenBSD 4.0

OpenBSD – оплот вашей безопасности… и все? **Майк Сондерс** решил разобраться.

Вкратце...

» Разновидность Unix, берущая начало в NetBSD и тяготеющая к вопросам безопасности малых серверов и брандмауэров. См. также: FreeBSD и дистрибутивы Linux.

Всего один удаленный эксплойт в стандартной инсталляции более чем за десять лет! Это впечатляющее достижение гордо обозначено на сайте OpenBSD, хотя данная формулировка вызвала известную неприязнь у пользователей других BSD и Linux. Действительно, стандартная инсталляция OpenBSD сильно ограничена и лишена изрядной доли функциональности – с тем же успехом можно назвать ультра-безопасной MS-DOS 1.0, просто из-за ее малых возможностей. А если серьезно, то вклад OpenBSD в укрепление компьютерной безопасности и в самом деле весом (вспомнить хотя бы пакет инструментов для удаленного входа OpenSSH), и нам захотелось посмотреть, что еще предлагает версия 4.0.

Чтобы финансово подпитать проект, лидер разработчиков Тео де Раадт [Theo de Raadt] решил не размещать ISO-образов CD онлайн; вместо этого они доступны по цене €50 за штуку, а нетерпеливые могут обратиться к FTP-инсталляции. Инсталлятор OpenBSD предполагает наличие предварительного опыта в Unix, действуя исключительно из командной строки без каких-либо меню, давно стоящих на вооружении инсталляторов NetBSD и FreeBSD. Оттого и умеренность системных требований – 16 Мб RAM вполне достаточно!



› **Версия 4.0 прочна и хорошо документирована, ничто новое не угрожает ее стабильности.**

> «Много новых драйверов для Gigabit Ethernet и беспроводных сетей.»

Меньше работы!

Основное новшество в инсталляторе – поддержка хост-варианта: теперь инсталлятор ищет на инсталляционном носителе архив под названием **site40-<hostname>.tgz** и, разыскав, устанавливает его. Это упрощает «нестандартную» инсталляцию с носителя или FTP и впоследствии требует меньше упражнений в командной строке.

Стандартная инсталляция не содержит *X Window System*, но некоторые фанаты используют OpenBSD в качестве настольной системы (хотя поддержка SMP и x86 не столь развита, как во FreeBSD и Linux). Доступно свыше 3400 бинарных пакетов, включая KDE 3.5.4, *Firefox 1.5.0.8* и *MPlayer 1.0 pre8*, поэтому настольная работа вполне возможна, при условии поддержки вашего оборудования. ПО, входящее в OpenBSD 4.0, весьма консервативно: *Apache 1.3.29*, *Sendmail 8.13.8* и *GCC 3.3.5* – версии, далекие от новейших; но все они снабжены заплатками от команды OpenBSD для повышения безопасности. ОС вовсю применяет разделение привилегий, *chroot* и случайное выделение памяти (то есть, автору эксплойта будет труднее выбрать «точку отсчета» для запуска shell-кода), и это дает администратору безопасную, хотя и минимальную, основу для строительства сервера или брандмауэра. Стандартная инсталляция, вместе с инструментами сопровождения, занимает примерно 400 Мб.

Большинство обновлений в версии 4.0 связаны с оборудованием: много новых драйверов для Gigabit Ethernet и беспроводных сетей, плюс улучшенная поддержка различных контроллеров жестких дисков. На платформенном фронте, теперь поддерживаются UltraSPARC III и Zaurus SL-C3200. К основному ПО добавлен OpenRCS, инструмент контроля версий под лицензией BSD, который заменил *GNU RCS* – замена GPL-лицензированного ПО на BSD продолжается.

Больше работы!

Некоторые аспекты ОС в сравнении с Linux смотрятся бледно. Каждая версия OpenBSD поддерживается лишь 12 месяцев. Неплохо, учитывая малочисленность команды, но это означает существенное увеличение нагрузки на администраторов по сравнению, скажем, с пятилетним циклом поддержки для серверов Ubuntu LTS.

Кроме того, поддержка и обновление системы неоправданно сложны и требуют перекомпиляции частей ОС из исходных текстов. Бинарные системы обновления (например, **apt-get upgrade**) давно стали нормой для Linux, а работать с ними гораздо быстрее и проще. Конечно, обновления в стиле OpenBSD дают полный контроль над любыми изменениями, к тому же компиляция проходит специально для вашей машины – но для вечно занятых администраторов это лишняя каторга. LXF

Linux Format Вердикт

OpenBSD 4.0

Разработчик: Команда OpenBSD
Сайт: www.openbsd.org
Цена: Бесплатно под лицензией BSD

Функциональность	7/10
Производительность	6/10
Простота использования	4/10
Документация	8/10

» *Все же неплохая ОС для малых серверов, заметно обновленная, но для облегчения работы администраторов можно было бы и постараться.*

Рейтинг 7/10

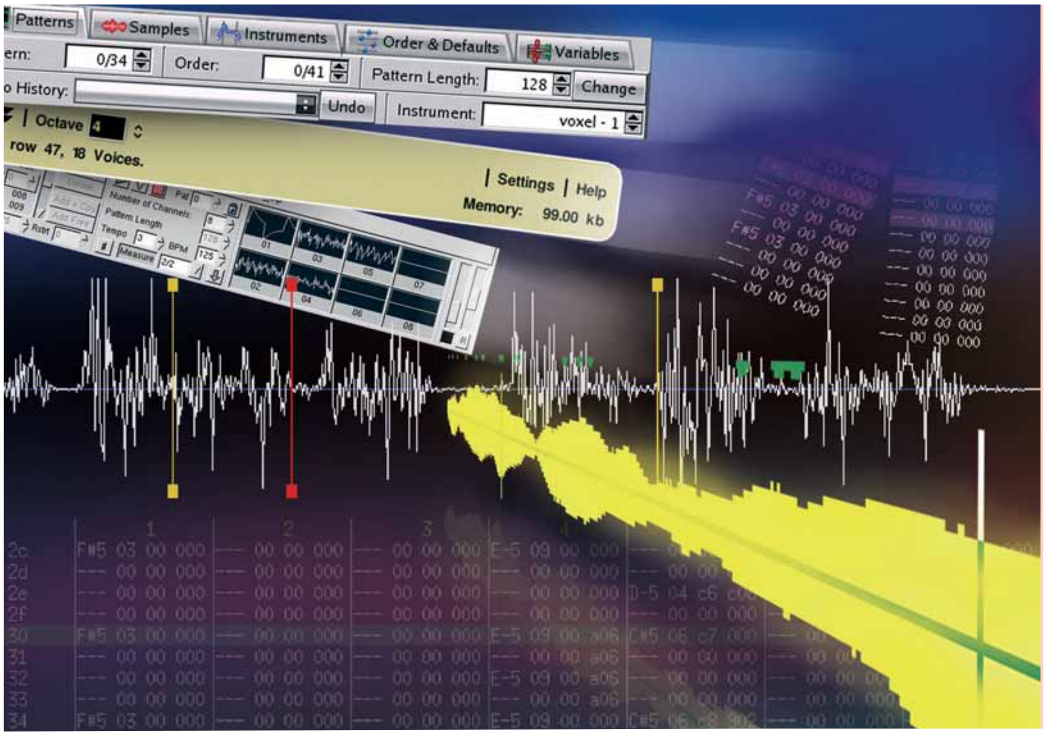
Сравнение

» Каждый месяц мы анализируем для вас тысячи программ – а вы можете отдохнуть!

LINUX FORMAT на DVD

Аудиотрекеры

The Ultimate SoundTracker породил новый жанр музыки. Двадцать лет спустя трекеры все еще сильны. **Грэм Моррисон** рассматривает шесть лучших из них.



Paul Blachford

Про наш тест...

Мы откопали несколько наших любимых «модов» из восьмидесятых и прогнали через тест шесть наиболее популярных трекеров для Linux. Вот на что мы обращали внимание:

Подлинность: Настоящему трекеру нужны: чрезвычайно сложный, псевдо-шестнадцатеричный редактор паттернов, редактор инструментов для создания сложных звуков и редактор сэмплов, преобразующий «сырой» шум в нечто пригодное для создания мелодии.

Совместимость: Несмотря на доверие к 20-летней технологии, идеальный трекер должен иметь современные ядро и экранное разрешение и работать с нашими аудиодрайверами ALSA без непременной установки престарелой Open Sound System (предтеча ALSA).

Простота использования: Мы искали дружественные инструменты запроса файлов, широкую совместимость форматов сэмплов и систему помощи, способную приоткрыть завесу тайны над темным искусством создания музыки.

Наш выбор...

Если вы раньше не слышали о модуле *SoundTracker*, вы, видимо, слишком юны, чтобы знать о нем. В конце восьмидесятых и на протяжении девяностых компьютерная музыка в основном распространялась и создавалась при помощи модуля *SoundTracker*, известного как «мод». Тогда на пересылку файла в несколько мегабайт уходило чуть ли не полжизни, и вся ваша коллекция музыки уместилась бы на одной 1,44Мб-дискете.

При тех скоростях передачи ледникового периода, решением было использовать программы, именуемые трекерами, создавая музыку с нуля при помощи последовательности кодов различных нот и эффектов, связывающих ноты с определенными сэмплами. Результатом был мод: законченный музыкальный файл со звуками или нотами, увязанными с данными аудио-сэмпла. Благодаря различным эффектам можно было создавать приличные музыкальные фрагменты, используя всего несколько коротких сэмплов, и многие из ранних модов не превышали 40 КБ.

Первый трекер такого типа назывался *The Ultimate SoundTracker*; написал его Карстен Обарски [Karsten Obarski] в 1987 г. для Commodore Amiga. Большинство современных программ-трекеров работают фактически подобно детищу Обарски: как правило, есть пять отдельных страниц для работы с файлами, редактирования паттернов, данных звуковых сэмплов, редактора инструментов и информации о композиции. Но в первую очередь большинство людей вспоминает именно его редактор паттернов. Он больше похож на электронную таблицу, чем на средство создания музыки, обычно с 64 строками для нот одна под другой и от четырех до 100 столбцов на каждую звуковую дорожку. Математически точная компоновка этого редактора паттернов придает особое звучание композиции, отчасти благодаря тому, что 64 вертикальных нотных позиции легко делятся на четыре (64/4 = 16 нот на каждый блок).

За последние 20 лет программы-трекеры сформировали для себя некоторую нишу. Их особый стиль программирования означает, что этот жанр никогда не умрет, так что аудиотрекеры сохраняют невероятную популярность. И в Linux доступна львиная доля хороших трекеров. Это **Сравнение** собирается выявить лучший из них.

SoundTracker

Начнем с самого начала.

Начнем с *SoundTracker*, потому что он и есть начало всему. Это прямая копия первой программы для музыкального трекинга (*The Ultimate SoundTracker*), породившей целый жанр зацикленной, повторяющейся танцевальной музыки, которая и принесла известность трекерам. *SoundTracker* воспроизводит наиболее близкие к оригиналу приемы работы с феноменом трекеров на вашей Linux-машине. Оборотная сторона – уж очень мало уступок XXI веку. *SoundTracker* выглядит какой-то червоточиной из 1987 в 2007 год.

Теперь о деле

Пользовательский интерфейс очень похож на оригинальный *SoundTracker* и разбивает окно приложения *GTK1* на три основных области. Данные о композиции засунуты в верхний левый угол, в правом верхнем углу – мониторинг звука, а всю нижнюю половину занимают пять вкладок, содержащих основную функциональность: управление файлами, редактор инструментов, редактирование сэмплов и данные модуля.

Эти функции одинаковы практически во всех трекерах, но страница File несколько удивляет – после 20 лет существования инструментов запроса файлов. Она работает по типу первоначального приложения восьмидесятых, с единственным окном просмотра файловой системы, используемым для загрузки и сохранения модулей, инструментов и аудиосэмплов. Вы просто щелкаете по типу файла, который хотите загрузить, затем переходите к месту, где он хранится.

Аудиосэмплы нужно преобразовать в монофонические WAV-файлы, и если вы попытаетесь загрузить стереофайл, то сможете выбрать либо левый, либо правый канал, или позволить *SoundTracker* смешать их на одном треке. Можно также загрузить «сырые» аудиоданные, но это полезно только при поиске данных сэмпла в каком-нибудь исполняемом файле. Большие аудиофайлы также вызовут проблемы. Лучший подход – создать библиотеку одиночных сэмплов и использовать аудиоредактор из *SoundTracker* для установки точек зацикливания и огибающих [envelope – контроль уровня звука, зачастую выражающийся графически, – прим. пер.]. Впрочем, это справедливо для каждого рассматриваемого здесь трекера.

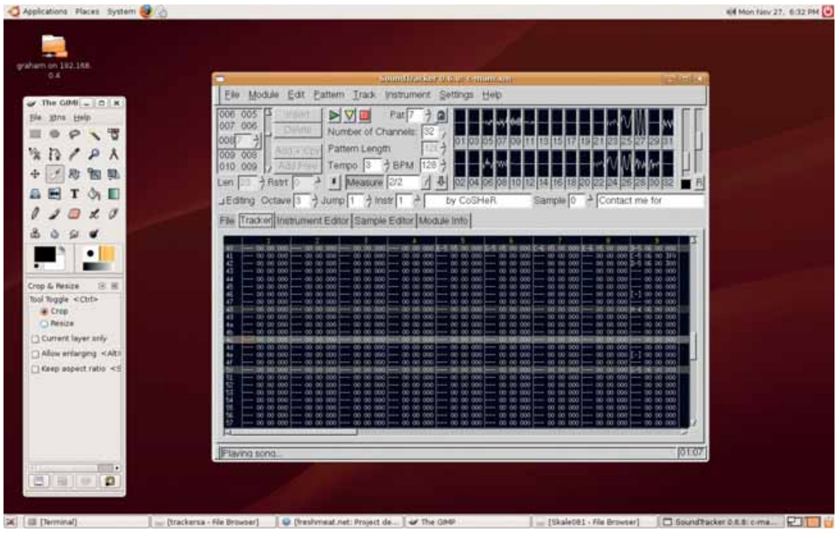
Следующий шаг – переход на вкладку Instrumental Editor. Отсюда можно менять громкость и высоту тона сэмпла и добавлять амплитудную огибающую для изменения громкости сэмпла во времени. Есть также возможность добавить вибрато, а прослушать любые выполненные изменения можно, «играя» на обычной QWERTY-клавиатуре.

Как только вы создадите инструмент или пару, действие перемещается в редактор паттернов. Это сердце любого аудиотрекера, и все они (в большей или меньшей степени) функционируют сходным образом. Однако компоновка *SoundTracker* понятна, и паттерн прокручивается вертикально, давая графическое представление текущей позиции воспроизведения. Это важно для редактирования неправильных нот, и более того, отметки, разграничивающие каждый такт и долю, привязываются к временной сигнатуре данной композиции – уникальная функция *SoundTracker*.

Спасибо за музыку

В верхнем левом углу экрана размещается область данных композиции, включая управление подачей (Play/Pause, Play Pattern и Stop), информацию о количестве ритм-ударов в минуту (BPM – beats per minute), длине паттерна и сопутствующих настройках. Наиболее важная часть этой области – список последовательности паттернов прямо под меню File. Создав новые паттерны, вы выстраиваете их последовательно, чтобы сформировать завершенную композицию путем повторения определенных кусков в определенное время. Кнопка Play Pattern очень полезна при создании собственных паттернов, поскольку не позволяет треку соскользнуть на проигрывание всей композиции.

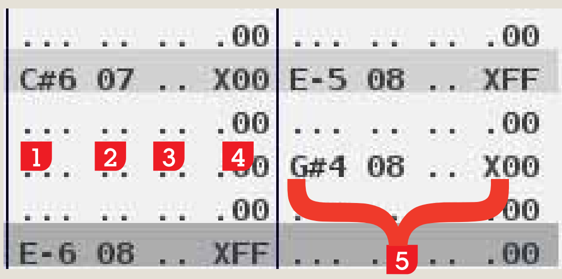
У *SoundTracker* нет современных украшений, и он приносит на рабочий стол Linux наиболее близкий к оригиналу способ работы. Старый дизайн графического интерфейса вынуждает вас создавать музыку особым способом, и именно поэтому так много людей любят писать музыку с помощью программ-трекеров. Было бы неплохо увидеть поддержку стереозвука, и, возможно, инструмент запроса файлов, но нас удивило, насколько этот метод создания музыки эффективен и насколько хороши результаты. *SoundTracker* – это, определенно, один из претендентов на победу, и несмотря на тот факт, что он уже несколько лет не обновлялся, будет интересно взглянуть, что сумели противопоставить классическому дизайну *SoundTracker* трекеры поновее.



› Классический вид трекера и паттернов *SoundTracker* почти не изменился за последние 20 лет.

Нотация редактора паттернов

1 Добавить ноту. Просто нажимайте клавиши.
2 Инструмент. Число, показывающее инструмент для каждой ноты.
3 Уровень звука. Чем больше, тем громче.
4 Эффект. Скольжение, дрожание и прочее.
5 Канал. Воспроизведение ограниченного числа каналов.



LINUX FORMAT Вердикт

SoundTracker

Версия: 0.6.8
Сайт: www.soundtracker.org
Цена: бесплатно под GPL

» *Вещь дельная, а к оригиналу просто ближе не бывает. Ему не хватает блеска, но для старта это превосходное место.*

Рейтинг 8/10

Schism Tracker

Продукт «старой школы»: крутой или просто старый?

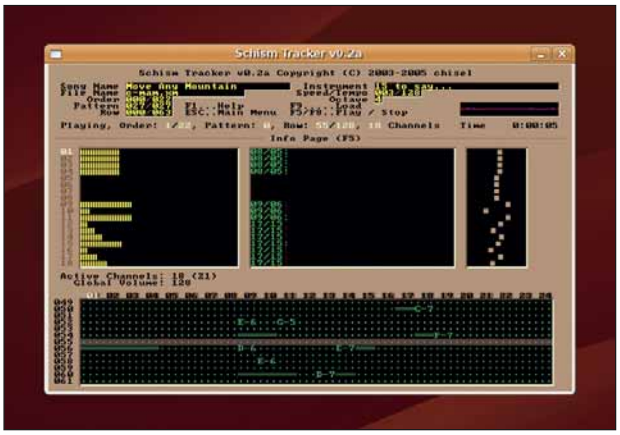
Schism Tracker (см. LXF87/88) тянет на награду в номинации «Самое странное имя», но вряд ли завоюет что-нибудь за дизайн пользовательского интерфейса. Вместо общепринятых органов управления, известных как «меню» и «кнопки», *Schism Tracker* использует исключительно комбинации клавиш. Почти все рассмотренные нами трекеры применяют такие же клавиши, унаследованные от оригинала, но это единственный трекер, воображающий, что пользователь способен все их запомнить.

Нажатие F5 воспроизведет вашу мелодию, а F8 остановит воспроизведение. F2 вызовет редактор/трекер паттернов, F3 и F4 откроют окна сэмплов и инструментов. Совершенно дурацкая выдумка – то, что *Schism* скрывает указатель мыши, оставляя вас в ожидании момента «прилива сил», чтобы дело пошло на лад. Впрочем, Ctrl+M вернет указатель на место.

Обратно в реальность

С другой стороны, все, что нужно от трекера, здесь есть – только упрятано за парой нажатий на клавиши. Но настоящая морока начинается при попытке создать собственный паттерн. Мы не смогли найти, есть ли здесь клавиатурные команды для очистки паттерна или хотя бы копирования и вставки оного. Добавление нот выливается в трудоемкий процесс переключения между окнами инструментов, редактором паттернов и страницей помощи для поиска всех этих важных клавиатурных команд. Только клавиша Escape поступилась принципами: она открывает удобное меню для переключения между функциями.

Больше всего впечатляет в *Schism* страница воспроизведения (F5, помните?). Если бы Стенли Кубрику [Stanley Kubrick – американский режиссер и продюсер, – прим. пер.] понадобилось в «Космической одиссее 2001» воспроизводить mod-файлы, то он заготовил бы такой же экран. Множество мерцающих огоньков и диаграмм уровня звука, пульсирующих в такт, в то время как биты из Pong съезжают то влево, то и вправо, согласно стерео-балансу. Но шарма «старой школы» недостаточно, чтобы спасти *Schism*. Слишком много в нем наворотов, и при работе он изрядно раздражает.



› **Добро пожаловать в 1987 год. Забудьте о компьютерной мыши: здесь она вам не поможет…**

LINUX FORMAT Вердикт

Schism Tracker

Версия: 0.2a
Сайт: http://regelseven.com/schism
Цена: бесплатно под GPL

» *Смотрелся бы круто где-нибудь в клубе за спиной ди-джея. Но в остальном от него толку мало.*

Рейтинг 3/10

ShakeTracker

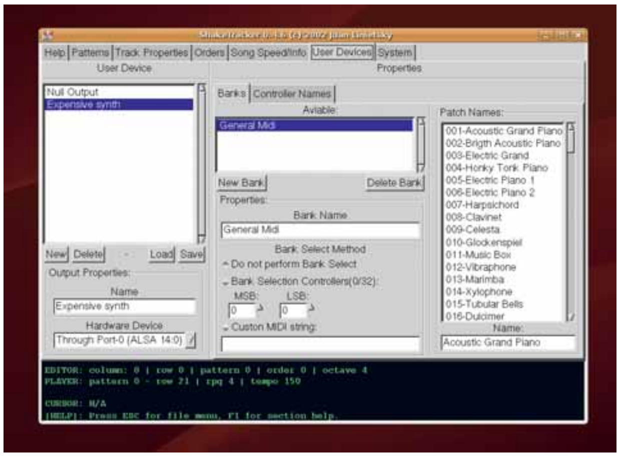
«I can't shake your love», поет Дебби Гибсон.

ShakeTracker заменяет страницы редактирования сэмплов и инструментов их MIDI-воспроизведением. Вместо паттернов, вызывающих внутренние инструменты, *ShakeTracker* отправляет ноты синтезаторам, подключенным на MIDI-порт. Это имело смысл в прежние времена, когда внешний синтезатор освобождал ценные ресурсы компьютера. Но в 2007-м оправдать такое сложновато.

Поэтому *ShakeTracker* будет полезен лишь тем, кому не обойтись без пользовательского интерфейса трекера для создания музыки и внешних синтезаторов для воспроизведения нот (или внутренних, типа программного синтезатора *Timidity*). *ShakeTracker* сопоставляет органы управления различным трекерным эффектам и наиболее общим инструментам, используя формат General MIDI. Пользовательский интерфейс ненамного изящнее, чем у среднего трекера, и нужно вручную создать новый трек, чтобы включить настройки своего MIDI-синтезатора, прежде чем вы сможете создавать и редактировать паттерны обычным способом. Каждый трек добавляется горизонтально, и редактирование музыки воспринимается так же, как в случае традиционного трекера. Ноты и эффекты набираются в паттерн, а композиции формируются путем добавления паттернов на страницу Orders.

MIDI-я

Результаты определенно интересны. *ShakeTracker* лучше всего работает в качестве хитрого арпеджиатора [arpeggiator – устройство для эмуляции аккордов за счет быстрого воспроизведения последовательности нот, – прим. пер.], повторяя простые музыкальные фрагменты с небольшим смещением, но это довольно далеко от создания музыки в понимании *SoundTracker*. Если у вас есть MIDI-оборудование, вам, может, и понравится работать в таком стесненном окружении – но огромным минусом будет то, что вы ни с кем не сможете поделиться своей работой, пока они не обзаведутся точно таким же MIDI-оборудованием. К счастью, вы можете экспортировать свою работу как MIDI-файл для использования с MIDI-секвенсером.



› **Из-за замены сэмплов на MIDI, страницы инструментов в *ShakeTracker* выглядят не как у всех трекеров.**

LINUX FORMAT Вердикт

ShakeTracker

Версия: 0.4.6
Сайт: www.reduz.com.ar/cheesetronic
Цена: бесплатно под GPL

» *Ох, ненамного больше удовольствия от ShakeTracker, чем от возврата на эстраду Дебби Гибсон.*

Рейтинг 4/10

ChibiTracker

Знать бы еще, что такое Chibi…

Решившие доказать, что трекеры пока живы и здоровы, взяли бы *ChibiTracker* за Экспонат Номер 1. Он все еще в активной разработке, последний релиз датирован концом ноября, и хотя пользовательский интерфейс следует знакомой формуле *SoundTracker*, он выглядит чуть изящнее и проще в использовании, чем большинство его собратьев.

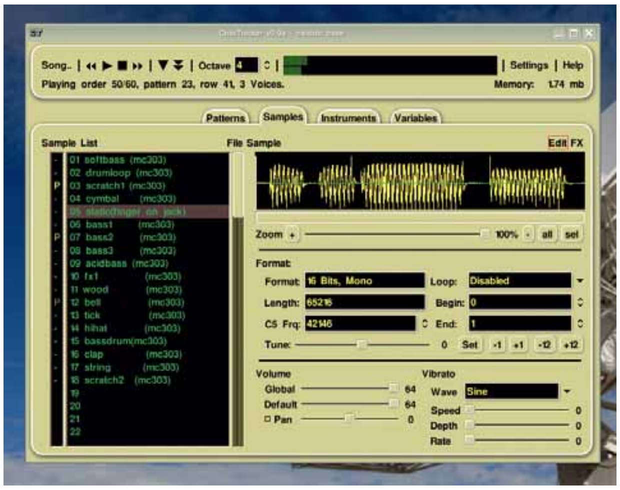
Он также укомплектован интерфейсом со сменяемыми темами (skins), поставляемым с несколькими готовыми, от аккуратной компоновки по умолчанию до различных имитаций ранних трекеров. Многофункциональную нижнюю половину основного окна можно переключать, используя четко обозначенные вкладки, а список паттернов содержит все 64 дорожки, выложенные на большую прокручиваемую панель. Вы можете точно выбрать, какие строки будут подсвечены, но временную сигнатуру вашей музыки, как в *SoundTracker*, изменять нельзя.

Окно паттернов позволяет также редактировать последовательность паттернов для создания композиции. Большинство других трекеров размещают редактирование порядка паттернов на отдельной странице, но это куда более логичное место для поисков. Редактор сэмплов сравнительно прост и без особых трудностей позволяет вам выделять фрагменты для вырезания и вставки. Замечательно, что он даже может редактировать сэмплы с поддержкой стерео.

Редактор инструментов также расположен очень логично: это самый простой для понимания редактор. Он обеспечивает амплитудные огибающие, панорамирование и органы управления высотой тона и фильтрами. Но лучшие дополнения можно найти на последней вкладке, помеченной как Variables. Именно здесь вы сможете добавить эффекты хора и реверберации [reverb – эффект запаздывающего звучания, – прим. пер.] ко всей композиции, и оба могут реально украсить вашу мелодию.

По-простому

Дизайн *ChibiTracker*, скорее, минималистичен. Нехватает прокрутки положения в композиции, как это сделано в *SoundTracker* – она просто уходит на дно паттерна, но мы сочли процесс сочинения музыки и воспроизведение простыми для освоения.



› ***ChibiTracker* умеет «менять лицо»: его можно состарить – или осовременить, как это сделали мы.**

LINUX FORMAT Вердикт

ChibiTracker

Версия: 0.9a
Сайт: www.chibitracker.com
Цена: бесплатно под GPL

» *ChibiTracker обещает многое, и это лучший выбор, если вы не пользовались трекерами раньше.*

Рейтинг 6/10

Skale

Трекер шикарный, но недоделанный.

Запуск *Skale* в первый раз – это как глоток свежего воздуха. Это приложение точно знает, кто такой типичный пользователь трекера, и не стыдится показывать свое происхождение, напоминая скорее старое «демо» Amiga, чем экземпляр музыкального ПО.

Но есть серьезная проблема: *Skale* закончен только наполовину. Все тут на месте, включая редактор паттернов, редактор звуков и редактор инструментов, и текущие модули воспроизводятся превосходно. Но, попытавшись использовать микшер или синтезатор, вы поймете, что *Skale* должен пройти еще долгий путь. При застопорившейся разработке (разработчики обещали новый релиз примерно раз в год, но никаких признаков дальнейшей активности нет), мы можем 20 лет дожидаться его завершения. И очень жаль, поскольку *Skale* – самый симпатичный на вид и самый интуитивно понятный трекер в нашем **Сравнении**. Одна только секция микшера способна резко изменить способы использования трекера, поскольку допускает произвольные эффекты и использование эквалайзера в реальном времени – если заработает. Многие страницы лишь наполовину выполняют обещанное, зияя отсутствующим функционалом.

Цифровая пианола

Тем не менее, можно создавать прекрасную музыку, используя традиционные страницы трекера: все они работают. Интерфейс пользователя – самый понятный из всех, которые мы пробовали, две группы кнопок посередине верхней части экрана открывают доступ ко всем основным функциям. И графика великолепная.

Редактор инструментов – прекрасный пример. Любовно прорисованная клавиатура фортепиано растянулась внизу экрана, и когда вы воспроизводите музыку из трекера, клавиши автоматически нажимаются, показывая, какие ноты звучат. Это трекер-эквивалент пианолы. Эх, кабы его доделали!

> «Интерфейс – самый понятный из всех.»



› **Кто знаком с разработчиками – дали бы им пинка! Пусть доделают все эти кнопки…**

LINUX FORMAT Вердикт

Skale

Версия: 0.8
Сайт: www.skale.org
Цена: бесплатно для скачивания

» *Это новое поколение программ-трекеров – ждем не дождемся его выхода из бета-статуса.*

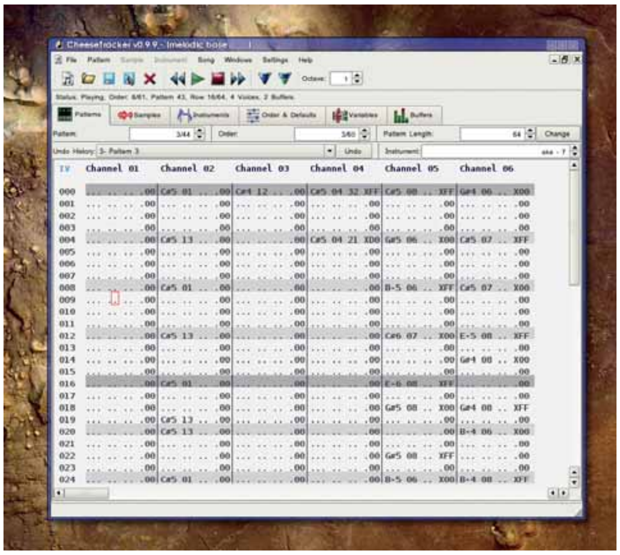
Рейтинг 6/10

CheeseTracker

Балансирует между уважением к прошлому и новыми технологиями.

CheeseTracker использует инструментарий *Qt*, и по виду и поведению больше походит на современное приложение, чем на персонаж демо-сцены восьмидесятых. У него те же компоненты, что и у других трекеров в нашем **Сравнении** – редактор паттернов, редактор сэмплов и редактор инструментов – но благодаря гибкости компоновочного движка *Qt* все масштабируется и адаптируется к разрешению вашего экрана.

Например, редактор паттернов работает именно так, как вы ожидаете. И есть несколько примет нынешнего века, доступных свободно благодаря *Qt*. Можно перетаскивать мышью выделенные фрагменты, выполнять копирование и вставку через обычный системный буфер обмена, есть даже буфер отката всех выполненных изменений. Это далеко ушло от угадывания клавиатурных команд; вы можете даже открыть отдельное окно, перечисляющее эффекты редактирования паттернов для ссылок на манипуляции со звуком в режиме реального времени.

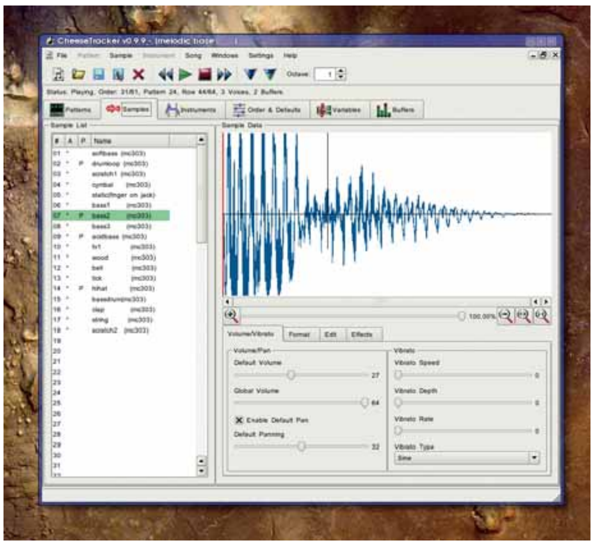


› В компании трекеров *CheeseTracker* проще всех: его функциональность упакована в интерфейс пользователя *Qt*.

Приложите руки

Компоновка в целом сохранила верность типажу трекера, используя вкладки для переключения между основными функциями. Если вы пользовались каким-нибудь из клонов *SoundTracker*, вы почувствуете себя комфортно и в *CheeseTracker*. Редакторы сэмплов и инструментов функционально идентичны таковым в *SoundTracker*, но гораздо проще в использовании, благодаря масштабируемому интерфейсу. Редакторы огибающих – лучшие из нами виденных, и они включают также эффекты фильтров. Но упорядочивание паттернов закинуто на страницу Order & Defaults – это своего рода чулан для различных функций, которые разработчик не смог обоснованно поместить в другое окно. Привыкнув к созданию композиции на странице редактора паттернов, как в *Skale*, не захочешь вот так скакать между страницами.

В окнах *CheeseTracker* можно найти интересные новшества для формата старых трекеров. Малоудачно названная вкладка Buffers содержит самые любопытные вещи. Эта страница работает подобно микшерной консоли, используя список из 16 каналов. Каждый канал реализует свою цепочку эффектов, что позволяет вам прогонять инструмент через цепочку внутренних эффектов или даже через любые эффекты LADSPA, которые вы установили (например, можно сопроводить задержку эффектом реверберации). Довольно просто сопоставить каждый инструмент с различными каналами в редакторе инструментов, чтобы применить цепочки эффектов к вашим звукам.



› Все трекеры позволяют редактировать сэмплы, но лишь *CheeseTracker* не превращает это в головную боль.

Можно также создавать собственные буферы и пропускать всю композицию через них, перед тем как отправить ее в основной канал вывода вашей звуковой карты. Применение эффектов действительно осовременивает идею *SoundTracker*, добавляя вашему шедевру изюминку. Вы не сможете поделиться ими с пользователями других трекеров, если у них не такая конфигурация, как у вас, хотя есть возможность выводить целый трек как аудио-файл и кодировать его как файл OGG. Вот чего нам действительно не хватало во всем приложении – это чтобы редактор паттернов отслеживал местонахождение в композиции: в *SoundTracker* это настолько полезно, что невольно ожидаешь того же в каждом приложении-трекере.

Джек-потуги

Еще одна функция, которая идет в ногу с обработкой эффектов – это поддержка Jack. То есть вы можете направлять выход с *CheeseTracker* в любое другое Jack-совместимое приложение – хороший кандидат, например, *Ardour*: вы могли бы записывать каждый паттерн трекера прямо на ваш инструмент звукозаписи, делая это частью более крупной композиции; скажем, *CheeseTracker* взять для треков ударных и баса, а *Ardour* – для добавления прочих треков. Именно эта гибкость дает *CheeseTracker* преимущество, когда речь заходит о совмещении трекерных композиций с современным окружением.

CheeseTracker не самая простая стартовая площадка, но этим инструментом стоит воспользоваться, если вы всерьез собрались создавать трекерную музыку, потому что он ведет себя как достижение последней пятилетки. Добавление буфера отката в окно паттернов – очень большое преимущество, а эффекты могут придать действительно профессиональное звучание вашей музыке.

> «Есть несколько интересных новшеств для формата старых трекеров.»

LINUX FORMAT Вердикт

CheeseTracker

Версия: 0.9.9
Сайт: www.reduz.com.ar/cheesetonic
Цена: бесплатно под GPL

» *Правильный выбор для трекероманов всего мира; достаточно гибок, чтобы не казаться неуместным на вашем рабочем столе.*

Рейтинг 7/10

Аудио-трекеры

Вердикт

SoundTracker 8/10

Другие трекеры лезли из кожи вон, чтобы превзойти это приложение, старающееся как можно полнее соответствовать исходным идеалам. Трекеры проектировались не затем, чтоб быть простыми в использовании или интуитивно понятными: создание музыки с помощью трекера требует скорее программирования, чем сочинительства, и это дает разработчикам трекеров повод избегать возни с помощью испытанных и проверенных формул.

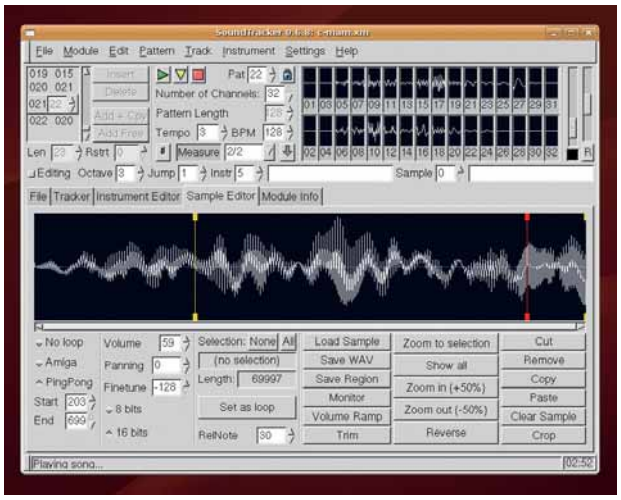
Но мы были бы рады видеть больше движения в сторону более удобного и современного дизайна интерфейса, и поэтому столь высоко оценили *CheeseTracker*. Он ближе всех из трекеров подошел к современному рабочему столу. Будущее жанра также выглядит блестящим со *Skale* и *ChibiTracker*. Если бы разработчики нашли время и потрудились чуть больше, оба этих новых трекера могли бы тягаться за корону *SoundTracker* уже через несколько месяцев (во всех новых версиях хотелось бы видеть более высокие частоты дискретизации и стереосэмплы).

С чувством реальности

Не выходя за пределы идеи, *SoundTracker* предлагает не только самый близкий к оригиналу способ работы, но и лучшую рабочую среду. Пользовательский интерфейс не менялся за 20 лет по уважительной причине: он переносит весь процесс написания музыки на кончики пальцев.

Приверженность *SoundTracker* первоначальному дизайну означает, что он использует все преимущества, накопленные за годы существования первоначального приложения. Его редактор паттернов был также самым простым в использовании, позволяя менять маркировку трека и следовать за указателем в композиции: обе эти вещи важны, если вы пишете много музыки. *SoundTracker* предлагает в основном те же способы работы, что и 20 лет назад, только вы уже не беспокоитесь о требованиях памяти или допустимом числе каналов – а для этого и нужно трекерное ПО.

Просто запустите это приложение, чтобы увидеть, что мы имеем в виду. Загрузите несколько старых классических мелодий с www.modarchive.com, побалуйтесь с инструментами в редакторе паттернов и сами создайте что-нибудь эдакое. Это прекрасно для поиска вдохновения и новых идей. Трекерная музыка обычно означала получение максимальной отдачи от ограниченного оборудования. В наши дни вы можете сами налагать ограничения на возможности и смотреть, как к ним адаптироваться. *SoundTracker* – это зрелый, отточенный инструмент для создания музыки, который наиболее близок к этой идее. Поэтому он и выиграл.



› *SoundTracker*, во многом воспроизводя Ultimate-прототип, выглядит реликтом, но как улучшить совершенство?

«Мы были бы рады видеть больше движения в сторону удобства и современного дизайна интерфейса.»

Ваше мнение

Мы знаем, что вы баловались с некоторыми из упомянутых трекеров. Если вы похожи на нас, то просто не сможете не оживить некоторых из тех старых воспоминаний и не набрать какую-нибудь случайную последовательность нот. Почему бы не отправить свою ретростальгическую попытку нам, чтобы она звучала в Башнях LXF? Пишите нам на letters@linuxformat.ru.

Таблица характеристик

	CheeseTracker	ChibiTracker	Schism Tracker	ShakeTracker	Skale	SoundTracker
ALSA [1]	✓	✓	✓	☒	✓	✓
OSS [1]	✓	☒	☒	☒	☒	✓
Jack [1]	✓	☒	☒	☒	☒	✓
MOD [2]	☒	☒	☒	☒	☒	✓
XM [2]	✓	✓	✓	☒	✓	✓
IT [2]	✓	✓	☒	☒	✓	☒
Стерео-сэмплы	☒	☒	☒	☒	✓	☒
Множественные проекты	✓	☒	☒	☒	☒	☒
Эффекты	✓	☒	☒	☒	✓	✓
Сглаживание звучания	✓	✓	☒	☒	☒	☒
Отслеживание позиции	☒	☒	☒	☒	☒	✓
Запись	✓	☒	☒	☒	☒	✓
MIDI	✓	☒	✓	✓	☒	✓

[1] Формат вывода, [2] Формат ввода

Distrowatch

Ежемесячная сводка новостей дистрибутивов Linux



ЛАДИСЛАВ БОДНАР
основатель, начальник, редактор и сотрудник DistroWatch.com.

GenКто?

Помните, как Gentoo Linux был любимцем среди дистрибутивов? Невозможно было начать обсуждение любого дистрибутива на форуме, чтобы кто-нибудь не превратил его в рекламу Gentoo. И вправду, Gentoo появился как подлинно уникальный проект, с четко определенными целями, прекрасной утилитой управления пакетами, обильной документацией и огромным, необъятным репозиторием программных пакетов. Gentoo Linux стал самым быстрорастущим дистрибутивом Linux всех времен.

Однако в последние годы проект впал в застой. Форумы пользователей забиты жалобами на плохое качество управления и рост числа ошибок, которые никто не собирается устранять. Блог разработчиков Gentoo, некогда форум для эффективной связи между разработчиками, часто служит местом излияния их желчи по поводу текущего положения дел Gentoo. И весь проект страдает от опасно высокой текучки разработчиков.

Нужны перемены

Как повернуть удачу лицом к Gentoo? Хорошим началом было бы приведение целей проекта в соответствие с желаниями конечного пользователя, а не разработчиков Gentoo. Строгий контроль качества и эффективные процедуры устранения ошибок также приветствуются.

Но самое важное, в чем Gentoo нуждается прямо сейчас, это сильный лидер, подобный тому, что уже был в лице Дэниела Роббинса [Daniel Robbins]. Это должен быть хакер с хорошей репутацией и солидным послужным списком, навыками решения споров между разработчиками и способностью направлять проект по четко определенному пути. Только тогда Gentoo сможет вернуть былую славу.

ladislav.bodnar@futurenet.co.uk

С музыкой!

64 Studio 1.0a Новый дистрибутив для творцов.

Мир разработки Linux породил невероятное число дистрибутивов, практически для всего, что можно вообразить. Один из последних новичков на этой переполненной сцене – 64 Studio. Как следует из названия, это дистрибутив для специалистов, призванный помочь творческим людям, создающим музыку, видео, графику и другие формы цифрового контента. Вначале разработчики хотели создать дистрибутив на базе Debian для высокопроизводительных процессоров AMD64, но потом откликнулись на просьбы пользователей и начали также сборку издания i386 64 Studio.



› **Новый 64 Studio 1.0 использует стандартный установщик Debian.**

Первое впечатление после установки 64 Studio – это огромное число приложений для музыкантов, доступных прямо из меню Gnome. Удивляться нечему: в конце концов, большинство творческих людей свободолюбивы, а потому естественно, что самые технически подкованные среди них сразу же приняли концепцию свободного ПО и GPL.

Центральное приложение здесь Jack, аудиосервер с низким временем отклика, созданный для обеспечения возможности перемещать звук между различными аудио-интерфейсами и программами обработки звука. От большинства проприетарных программ Jack отличается тем, что не приковывает пользователя к одному из множества конкурирующих стандартов, а позволяет любому аудио-интерфейсу и приложению легко работать вместе. Приятным дополнением к Jack-серверу является *Jack Control* – графический интерфейс для настройки различных параметров.

Парад аудиоприложений

Среди доступных аудиоприложений – *AmSynth, Ardour, Audacity, QSynth* и *Rosegarden*. Имеется также несколько утилит управления воспроизведением и громкостью, и набор инструментов для работы с устройствами, например, PCI-картой VIA Vinyl Envy24 PCI. И много, много чего еще – новообращенный пользователь аудиоприложений с открытым кодом может просто оторопеть, увидев такое изобилие имеющегося разнообразного ПО, и все бесплатно!

Хотя 64 Studio сфокусирован на аудиопотребностях, здесь также представлены разнообразные приложения для работы с видео (*Kino, Totem*) или графики (*Blender, Gimp, Inkscape*), наряду со стандартным ПО для web (Firefox), электронной почты (*Thunderbird*) и простой офисной работы (*AbiWord, Gnumeric, Scribus*). Фактически, это дистрибутив, способный заменить недешевый набор проприетарных приложений для творческих людей или, в худшем случае, составить компанию основной ОС музыканта при двойной загрузке.

www.64studio.com



› **Большой выбор приложений для музыкантов включает и эту драм-машину *Hydrogen*.**

Конничи-ва, Тux!

Vine Linux 4.0 ОС азиатского сообщества.

Япония была одной из первых стран вне Северной Америки и Европы, которая сказала «Здравствуй, Тукс!». Вначале это был проект Japanese Extension, с целью добавить поддержку японского языка в Red Hat Linux, но позднее, в 1998, ведущие разработчики проекта решили клонировать Red Hat и создать свой собственный дистрибутив под именем Vine Linux. Теперь это наиболее популярный дистрибутив сообщества Linux в Японии.

Vine Linux 4.0 – первый выпуск проекта, использующий ядро 2.6, что сильно расширяет поддержку оборудования его пользователям. Рабочим столом выбран Gnome 2.14, с улучшенным набором последних GTK-приложений, например, медиапроигрывателями *Beep* и *Totem*. Японским методом ввода по умолчанию теперь является *Anthy*, вытеснивший предыдущего фаворита *Canna*. Как и все предыдущие релизы Vine Linux, версия 4.0 использует систему установки Anaconda от Red Hat, но управление RPM-пакетами осуществляется Debian'овскими *APT* и *Synaptic*.



› Дайсуке Судзуки [Daisuke Suzuki], основатель проекта Vine.

По умолчанию используется кодировка eucJP, но Vine Linux 4.0 также поддерживает английский (UK или American), а переключать языки очень просто. Понятно, что дистрибутив в первую очередь интересен тем пользователям Linux, кому необходим японский, но даже если вам неохота вникать в сложности этого языка, Vine Linux все же солидный и интересный продукт, заслуживающий внимания.
www.vinelinux.org

Дистрибутив за $100

Xandros Desktop Professional Новый: 3D!

Корпорация Xandros уже несколько лет поставляет дружественные к пользователю решения Linux для дома и офиса. Хотя Linux-сообщество часто критикует ее за отсутствие обратного вклада в Debian и другие открытые проекты, формирующие основу коммерческого предложения, ее дистрибутив Linux часто хвалят как одну из наиболее понятных замен Windows, существующих на рынке. В ноябре 2006 компания выпустила свой новый продукт: Xandros Desktop 4.1, продаваемый как Xandros Desktop Professional.

Новая версия, на базе несколько устаревшего кода Debian 3.1, поставляется с набором интересных обновлений. Последнее ядро 2.6.18 с соответствующими обновлениями для графических карт ATI и Nvidia, предоставляет улучшенное обнаружение устройств для современных настольных систем; обновлены также популярный браузер *Firefox* (2.0) и коммерческий пакет *CrossOver Linux* (5.9.1). Улучшенная поддержка доступна на чтение и запись к разделам Microsoft NTFS – еще одно новшество. И, вслед



› Настольный инструмент поиска *Beagle* – одно из нововведений в Xandros.

другим популярным дистрибутивам, 3D-эффекты рабочего стола, благодаря *Compiz* и *Xgl*, тоже дебютировали в этом релизе.

Xandros Desktop Professional продается по цене 100$, но не содержит никаких серьезных инноваций – кроме того, большинство озвученных новых функций имеются также и в других системах – но это солидный дистрибутив для пользователей, которые ценят свое время и хотят, чтобы компьютер помогал в решении их задач, а не был чем-то вроде хобби. LXF
www.xandros.com

Передний край

В отличие от издателей приложений с закрытым кодом, многие дистрибутивы Linux поддерживают общедоступные репозитории программ с текущим снимком разработки (snapshot). По большей части они довольно стабильны, но могут страдать от второстепенных, а иногда и серьезных ошибок после загрузки в репозиторий программных пакетов отдельными разработчиками. Когда дистрибутив переходит в стадию общественного тестирования (и текущий снимок выпускается как альфа, бета, релиз-кандидат или тестовый выпуск), древо разработки медленно стабилизируется, загрузка пакетов запрещается, и ветка входит в стадию «заморозки». После недель отладки, дерево объявляется стабильным и выпускается для потребителя. Ниже приведен список древ разработки основных дистрибутивов:



› Ветка разработки Debian Sid также известна как Unstable.

Дистрибутив	Название	Комментарий
Debian	Sid	Имеются также ветви Testing и Experimental.
Fedora	Rawhide	Разработчики Fedora и старые Red Hat'овцы все еще используют этот термин.
Gentoo	Unstable	Набор метафайлов со ссылками на пакеты с исходным кодом.
Mandriva	Cooker	Это имя придумал Гаэль Дюваль, и оно так и приклеилось.
OpenSUSE	Factory	На заре OpenSUSE называлось Edge, позднее переименовали.
Slackware	Current	Имеет тенденцию обновляться редко, но помногу.
Ubuntu	Feisty	Имя соответствует будущему релизу, сейчас это Feisty.

Хит-парад дистрибутивов

10 самых посещаемых страниц на DistroWatch.com с 9 ноября по 8 декабря 2006 (среднее число визитов в день)

	Дистрибутив	Число визитов	
1	Ubuntu	2,299	‹›
2	SUSE	2,168	‹›
3	Fedora Core	1,459	‹›
4	Mint	1,069	↑
5	SimplyMepis	1,040	‹›
6	Debian GNU/Linux	1,004	‹›
7	Mandriva	916	↓
8	PCLinuxOS	903	↓
9	Damn Small Linux	902	↓
10	Slackware	612	↓

» DistroWatch.com следит за популярностью дистрибутивов, основываясь на числе визитов на страницу каждого дистрибутива. Хотя оно не соответствует реальному числу установок, но показывает, какие дистрибутивы более популярны за определенный промежуток времени.



MONO уже с нами

Самая крутая вещь в Windows ныне стала самой крутой и в Linux. И пользователь, и разработчик, и администратор что-нибудь да найдет в Mono. Рассказывает **Пол Хадсон**.

Вначале был .NET – кросс-платформенная система для разработки высокоскоростных программ, и было это хорошо. На беду, он был проприетарным ПО от Microsoft, так что Святой Столлмен считал его дурным, и в мире свободного ПО его избегали. Затем Бог прислал спасителя в форме Mono: тот, кто программирует на Mono, не привязан к одной платформе, но имеет неизменную производительность и неизменный почет среди пользователей открытых программ.

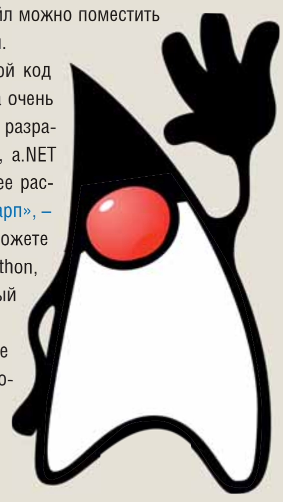
Будь вы пользователем настольной системы или программистом, Mono призван изменить работу с вашим компьютером. Он уже применяется в четырех самых интересных приложениях Linux, и еще больше их на подходе. Если вы разработчик, вы обнаружите, что Mono проводит вас от прототипа до законченного продукта гораздо быстрее, чем более устоявшиеся языки программирования, вроде C, C++, Java или Python.

Но что есть Mono, и почему люди так влюблены в него? Читайте дальше – узнаете...

.NET против Java

Кросс-платформенная природа .NET стала возможной, потому что программы компилировались в специальную форму, известную как универсальный промежуточный язык (CIL, common intermediate language), не привязанный к конкретному процессору. Когда программа запускается первый раз, Mono конвертирует CIL-код в оптимизированный машинный код для процессора, на котором он работает, и это объясняет, как один и тот же исполняемый файл можно поместить на какую угодно машину и оптимизировать для любой ситуации.

Вероятно, это похоже на Java, где разработчики пишут свой код единожды, и он может работать везде. И правда – .NET и Java очень похожи. Различие в том, что платформа Java была изначально разработана для использования с языком программирования Java, a.NET проектировался для использования с любым языком. Наиболее распространенный .NET-язык называется C# [произносится «си-шарп», – прим. ред], и он во многом основан на Java и C++. Но вы можете также использовать Visual Basic, C++, Cobol, Eiffel, Perl, PHP, Python, Ruby, Tcl и даже Java – все они компилируются в тот же самый CIL-код, и полностью совместимы.



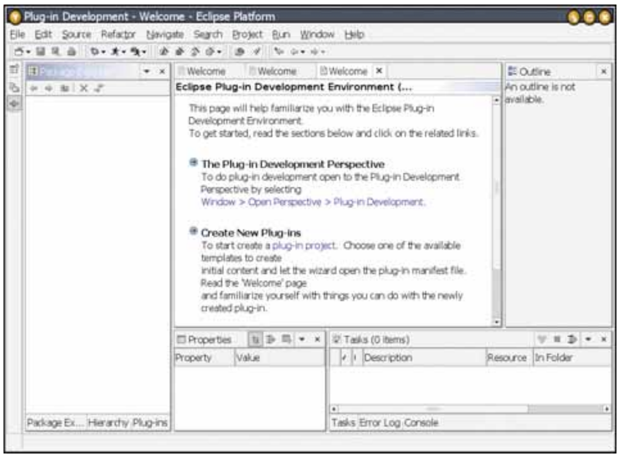
Значит, над одним проектом сможет работать команда, где каждый программист пишет на своем любимом языке: PHP-программист может строить что-то поверх кодовой базы C#, созданной кем-то другим и унаследованной от некоторого кода Visual Basic, написанного еще кем-то.

И это все о .NET

Если вы не слышали раньше имени Мигеля де Икасы [Miguel de Icaza], самое время с ним познакомиться. Это супер-хакер из Мексики, прославившийся как основатель проекта Gnome, создатель *Gnumeric*, со-основатель фирмы, со временем ставшей Ximian и выпустившей *Evolution*. Он также основал проект Mono: движение по переносу Microsoft .NET в мир свободного ПО.

Конечно, это подводит к вопросу: а что же такое .NET? По-простому, это система программирования, позволяющая создавать программы очень быстро и легко. Например, она включает библиотеки для работы с XML, графическими пользовательскими интерфейсами, безопасностью и с другими стандартными задачами. Более того, она автоматически управляет выделением памяти, так что разработчикам не нужно следить за каждым используемым байтом, как в былые времена.

Но важнее всего в .NET ее кросс-платформенность: вы можете взять исполняемую .NET-программу и запустить ее на Windows, как хотелось Microsoft, или использовать Mono, чтобы заставить ее работать на Linux, BSD, Mac OS X и многих других платформах. Вы также можете запускать программу на 32- или 64-разрядном оборудовании, и она будет динамически оптимизирована для максимально быстрой работы на данных машинах. Как пользователю Linux, вам следует уже понять, почему Mono пробуждает интерес такого количества людей: она исключает многие, столь распространенные проблемы зависимостей. Вам не нужно беспокоиться об установке *libfoo.so.1.2.3.4-5*, не нужно волноваться, будет ли программа работать на вашем Athlon64, и даже не нужно ничего компилировать самому – просто скачайте уже кем-то скомпилированную версию, и она заработает на вашей машине.



› **Мигель де Икаса сделал этот снимок экрана *Eclipse*, запущенного в Mono, несколько лет назад, для показа скорости развития Mono.**



Mainsoft

› **Мигель вкалывает над Mono 2.0. Или занят игрой в *Tux Racer*.**

На сегодняшний день Microsoft выпустила четыре версии .NET, под номерами 1.0, 1.1, 2.0 и 3.0. Каждая из них имеет обратную и прямую совместимость с другими, означающую, что программа, написанная для .NET 1.0, будет работать и на .NET 2.0, а если не использовать код, специфический для .NET 2.0 – программа, написанная для .NET 2.0, будет работать на .NET 1.1. На практике, это отличный подарок пользователям: если у вас последняя версия Mono, вы можете запускать почти все без оглядки на номера версий.

Сейчас Mono полностью поддерживает .NET 1.0 и 1.1 и многое из 2.0. Сюда включена поддержка для *Windows Forms 1.1*, графического инструментария .NET. Это значит, что вы можете взять сотни Windows-программ, даже не подозревающих о существовании Mono, и запустить их в Linux. Если вы думаете, что это выглядит как *Wine*, вы почти

> «Mono исключает многие, столь распространенные проблемы зависимостей.»

правы. Но Mono добавляет гарантированную совместимость, родную поддержку для многих других архитектур (*Wine* поддерживает только 32-разрядные процессоры семейства x86) и увеличенную скорость благодаря оптимизации именно на вашем оборудовании.

Mono в вашем дистрибутиве

Поскольку Mono основан на технологии Microsoft, при его представлении сообществу многие вслух опасались, что судебный иск к Mono – это лишь вопрос времени. Действительно, Microsoft в 2003 г. получила патент США 20030028685, фактически охватывающий весь каркас .NET Framework.

Но к тому времени разработка Mono была уже в процессе (сначала в Ximian, а затем в Novell, которая купила эту фирму в августе), и в 2003-м Mono уже был способен запускать *Eclipse*, используя специальный слой совместимости с Java, известный как IKVM. Позиция команды Mono по этим патентам всегда была простой: пока можно обходить патент, сохраняя совместимость, так и следует поступать. Если этого сделать нельзя, код помечается как «не реализовано», и начинается поиск prior art (кода, который был опубликован раньше запатентованного), лишающего этот патент законной силы.

«Из всех приложений, работающих на Linux, Mono доставляет меньше всего проблем – нам просто довелось принять на себя львиную долю критики, – говорит де Икаса. – Проекты типа *Mozilla* и *OpenOffice.org* годами копировали инфраструктуры (в форме XPCOM в *Mozilla* или UNO в *OpenOffice.org*), но там никогда не вставал этот вопрос».

Немалая часть .NET стандартизирована ISO – она и формирует основу Mono. Даже сейчас, когда Novell и Microsoft пришли к особому пониманию по использованию патентов (см. Новости в LXF86), разработчики Mono по-прежнему придерживаются своего плана избегать использования запатентованных технологий.

Поскольку Mono – проект от Novell, SUSE был первым крупным дистрибутивом, поставляемым с поддержкой Mono прямо «из коробки», но с тех пор Fedora Core и Ubuntu тоже включили ее в свой состав. Хотя .NET первоначально создавался для использования только в Windows, Mono расширил ее поддержкой десятков популярных в Linux библиотек и программ, включая *GStreamer*, *Avahi*, *Evolution*, *GTK* и *Cairo*.

По существу, в Mono взяли проверенную временем стратегию Microsoft «Заимствуй и расширяй», и для разнообразия приложили ее в наших интересах. Так что .NET больше не ориентирована только на запуск Windows-приложений на других платформах – Mono дает программистам возможность писать оригинальные приложения для Linux, при этом сполна используя преимущества .NET.

В этом спецрепортаже мы рассмотрим подборку приложений Mono, коренным образом изменивших рабочий стол Linux, и узнаем от ключевых разработчиков, что день грядущий нам готовит... »

Microsoft про Mono

Когда мы спросили Microsoft об их взглядах на Mono, вот что они ответили: «Mono – показатель восхищения .NET сообществом разработчиков... Притом, Mono – это попытка Novell получить части Microsoft .NET Framework методом инженерного анализа. Это не расширение .NET Framework, и не должно считаться таковым.»

Настольная одиссея

Первые плоды Mono все хорошеют. Попробуйте их прямо сейчас!

Beagle

Мгновенный поиск файлов – задача непростая, но выгода для пользователей огромная...



Джо считает...

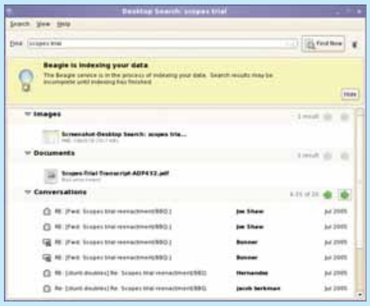
«Наша задача – индексировать всю информацию пользователя, и если мы пропустили какие-то типы файлов, мы это исправим.»

Что, если бы ваш компьютер определял, над чем вы работаете, и динамически показывал бы вам всю информацию, связанную с текущим заданием? Эта мечта легла в основу проекта *Dashboard* Ната Фридмана, и чтобы ее достичь, разработчикам нужно было создать систему, умеющую читать файлы в самых различных форматах и преобразовывать их в базу данных, допускающую поиск. Так и появился *Beagle*: он принимает запрос на поиск, а затем ищет все файлы, ему соответствующие. Документы, сообщения электронной почты, рисунки, журналы чатов, история web-браузера, новости RSS и прочее – *Beagle* все может прочитать, а значит, вы можете увидеть результаты из всех этих источников за один проход. Это произошло в 2004 г., и с тех пор *Beagle* стал одним из популярнейших проектов Mono.

Но дорога от *Beagle* 2004 года к сегодняшнему не была гладкой. «Я думаю, мы просто не представляли всех трудностей, когда начали разработку *Beagle*», говорит Джо Шоу [Joe Shaw], ведущий разработчик *Beagle*. «Поиск в настольной системе – невероятно сложная проблема, куда сложнее web-поиска, и труднее всего держаться на уровне новшеств по мере их появления, при этом не влияя на привычную работу пользователей с настольными системами».

Сейчас *Beagle* – уже не Gnome-ориентированный проект, каким был раньше: он читает файлы *OpenOffice.org*, журналы Kopete IRC, адресные книги *Thunderbird*, фильмы *QuickTime*, RPM-файлы, исходные тексты Python и многое, многое другое. И команда постоянно работает над расширением поддержки других типов файлов: это одна из областей, где *Beagle* кладет на обе лопатки *Spotlight* от Apple, во многом решающий ту же задачу. Как говорит Шоу, «Например, новая программа mind mapping под названием *Labyrinth*, появилась несколько месяцев назад, и спустя неделю после ее первого релиза *Beagle* уже распаковывал и индексировал ее файлы. В том-то и сила Open Source».

Одно из горячих будущих направлений разработки – способность сканировать архивы: tar и zip-файлы. Сейчас *Beagle* сканирует файлы архивов и сохраняет имена содержащихся в них файлов, но работа ведется в направлении индексации также и содержимого архивов: если у вас есть zip-файл с приобретенными файлами MP3, *Beagle* прочитает их имена, а заодно и ID3-тэги для этой музыки. Вы сможете выполнять поиск по имени певца, названию альбома и так далее.



› Наберите запрос в этом окне, и *Beagle* просканирует ваши файлы, почту, чаты и прочее – собака-ищейка отдыхает!

F-Spot

Фотоальбомы – это хорошо. Мы спросили, что привносит в них Mono.



Ларри считает...

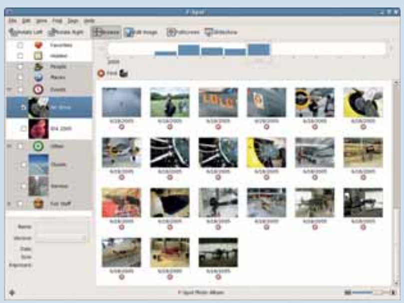
«Я хорошо знаком с GEGL и рад процессу, который наконец-то пошел. Мы обдумываем использование GEGL в *F-Spot*, но конкретных планов на сей счет пока нет.»

В последние годы любительская фотография снова активизировалась, благодаря появлению цифровых камер. Сегодня даже мобильные телефоны комплектуются камерами приличного качества, а значит, возникает серьезная потребность в организации, редактировании и печати фотографий, наряду со способностью нарезать их на CD-диски или выгружать в Интернет. Именно эту потребность и стремится удовлетворить *F-Spot*, уже являющийся de facto инструментом управления фотоальбомами в Linux.

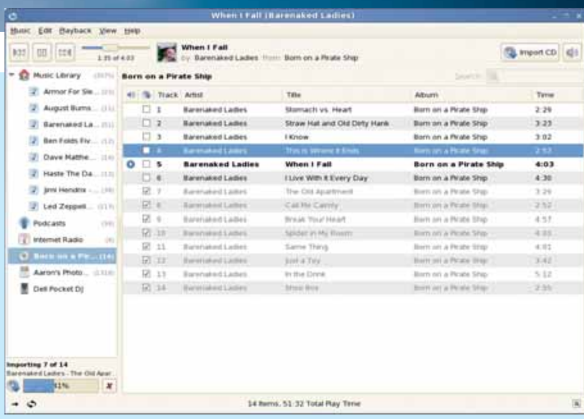
Ларри Юинг [Larry Ewing], больше известный как создатель Тукса и участник проекта *Gimp*, отвечает за *F-Spot*, и он не колеблясь поддерживает платформу Mono. «Mono предлагает простые способы приблизиться к оборудованию, так что значительная часть кода обработки изображений, требовательная к производительности, сосредоточена в библиотеках C, а логика приложений сохраняется полностью управляемой. Если бы мне пришлось начать ту же задачу с нуля, я снова выбрал бы Mono.»

F-Spot – одно из немногих Linux-приложений, предлагающих «родную» поддержку фотографам, получающим свои фотографии в RAW-формате. Это необработанный несжатый вывод с матрицы камеры, обеспечивающий наибольшую детализацию. *F-Spot* уже читает RAW-формат всех популярных моделей камер, и Юинг предсказывает, что к середине следующего года он будет расширен, так что фотографы смогут еще и редактировать свои RAW-изображения. Редактирование образов в других форматах (большинство камер используют JPEG или TIFF), уже реализовано – можно вращать изображения, менять их размер, обрезать, корректировать цвета и четкость; многие функции (включая мягкий фокус и коррекцию перспективы) уже запланированы в следующий основной релиз.

Разовьется ли *F-Spot* в долгосрочной перспективе в приложение, подобное *iPhoto*? Юинг принимает это близко к сердцу: «Долгосрочная цель – сделать больше и сделать лучше. Когда это будет, и будет ли, зависит от того, насколько большое сообщество сможет работать над *F-Spot* и как хорошо я работал до сих пор... Не люблю говорить о планировании номеров версий, так что вместо этого скажу, что *F-Spot* в SUSE Linux Enterprise Desktop 11 будет знаменовать собой конец большого цикла разработки.»



› В *F-Spot* изображения можно помечать тэгами и группировать, а можно просматривать по дате создания, с помощью умного слайдера вверху.



› **Вставьте аудио-CD, и *Banshee* скачает названия треков и скопирует их на ваш жесткий диск – а заодно и воспроизведет!**

Banshee

Медиа-плейер «все в одном», открывший Linux для MP3.



Аарон считает...

«Одна из моих любимых долгосрочных задач – более тесная интеграция с web-сервисами типа Last.fm. Например, когда я слушаю песню, я хочу знать, когда этот певец будет выступать где-нибудь в наших краях».

Одни программы извлекают аудио-данные с CD, другие – воспроизводят MP3, а третьи – подключаются к iPod. Но *Banshee* уникальна тем, что выполняет все три эти вещи. По словам Аарона Боковера [Aaron Bockover], ведущего разработчика *Banshee* в Novell, *Banshee* преуспевает, потому что «здесь есть ряд основных функций, которых просто нет у других медиа-плейеров. Например, полностью интегрированные копирование и прожиг CD – мы не зависим ни от какой внешней программы для выполнения этой работы, что здорово улучшает пользовательские характеристики продукта».

Успех *Banshee* обеспечили два главных компонента: возможность написания сторонних расширений и превосходная поддержка iPod и других цифровых аудио-плейеров. Возможность писать расширения для ПО – вещь не новая, но *Banshee* специально разработан под либеральной лицензией MIT/X11, чтобы Novell могла встроить в SUSE Linux Enterprise Desktop 10 поддержку проприетарного кодека (т.е. MP3). «Раньше этого не мог сделать ни один поставщик из-за проблем с патентами и лицензиями», говорит Боковер. «Другие медиа-плейеры нельзя собирать с поддержкой закрытых кодеков, их лицензии чересчур ограничивающие».

Поддержка различных MP3-плейеров также реализуется через расширения *Banshee*, но на этот раз основное преимущество – необычайная простота присоединения к разработке членов сообщества. В конечном счете продукт, который волнует большинство – это iPod, а значит, *Banshee* должна идти в ногу с огромным множеством различных моделей iPod. Такая прагматичная позиция доводить дело до конца означает, что *Banshee* уже популярен среди владельцев iPod, а это основная масса обладателей MP3-плейеров, как хорошо известно Боковеру. «Хотя это закрытый и проприетарный продукт, он владеет где-то около 80% рынка цифровых аудио-плейеров».

Через несколько месяцев появится новая трековая модель и виджет просмотра, спроектированный для повышения производительности и сокращения потребления памяти. Повышение производительности всегда приветствуется, но Боковер говорит, что оно еще и «открывает дверь действительно отличным украшениям и новым функциям, например, группировке альбомов». Что ж, поглядим...

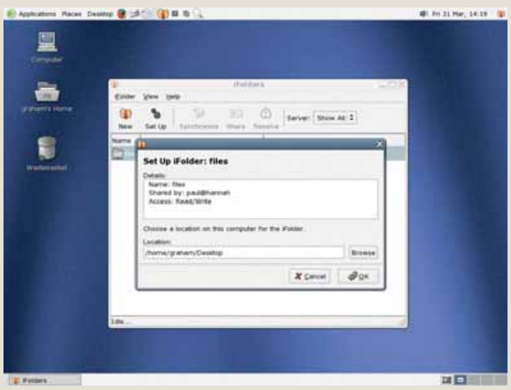
Также на подходе...

Mono доказывает, что он – удивительный хит в индустрии игр, где по меньшей мере три крупных проекта пользуются открытой природой Mono.

Первая и, вероятно, наиболее известная – притягивающая аватары *Second Life*, находящаяся в процессе переноса своего скриптового движка на Mono. Ее создатели уже обнаружили, что созданные пользователями скрипты выполняются в 50–150 раз быстрее, чем на старом Linden Script Language (LSL), а также ухитряются потреблять вдвое меньше памяти. Для упрощения трансляции они написали собственный LSL-компилятор для Mono, позволяющий сохранить существующие скрипты, а Mono открывает возможность к использованию всех языков, доступных для .NET, например, C#.

Наряду с *Second Life*, есть также проект Tao, портировавший OpenGL, SDL и другие игровые библиотеки на .NET; и движок *Unity 3D* – это закрытый инструментарий разработки игр, использующий Mono для работы на Windows, Linux и Mac OS X.

Еще три приложения подают голос на рабочем столе: *Tomboy*, *Blam* и *iFolder*. Мы уже рассказывали о *Tomboy*, но он продолжает хорошеть: наша любимая новая функция – способность перетаскивать электронные письма из *Evolution* в ваши *Tomboy*-заметки, создавая специальную ссылку, по которой можно щелкнуть, чтобы открыть письмо в будущем. *Blam* – набирающий популярность RSS-клиент, недавно сменивший своего ответственного: новый разработчик, Карлос Мартин [Carlos Martin], уже работает над *Blam 2.0* и собирается добавить поддержку web-сервисов, а также поддержку тэгов и фильтрацию новостей. Наконец, мы уже рассматривали *iFolder* в **LXF80**, но работа продолжается и здесь – в следующую версию, *iFolder 3.6*, планируется включить поддержку шифрования на лету, а также репликацию «master-slave» для обеспечения крупномасштабной синхронизации.



› **iFolder синхронизирует файлы между компьютерами с помощью протокола обнаружения сервисов Bonjour от Apple.**

Дополнительная информация

» http://beagle-project.org *Beagle*: поиск в настольной системе с оболочкой Gnome, но не только для пользователей Gnome...

» http://en.opensuse.org/Kerry ...это домашняя страничка Kerry – оболочки к *Beagle* для KDE.

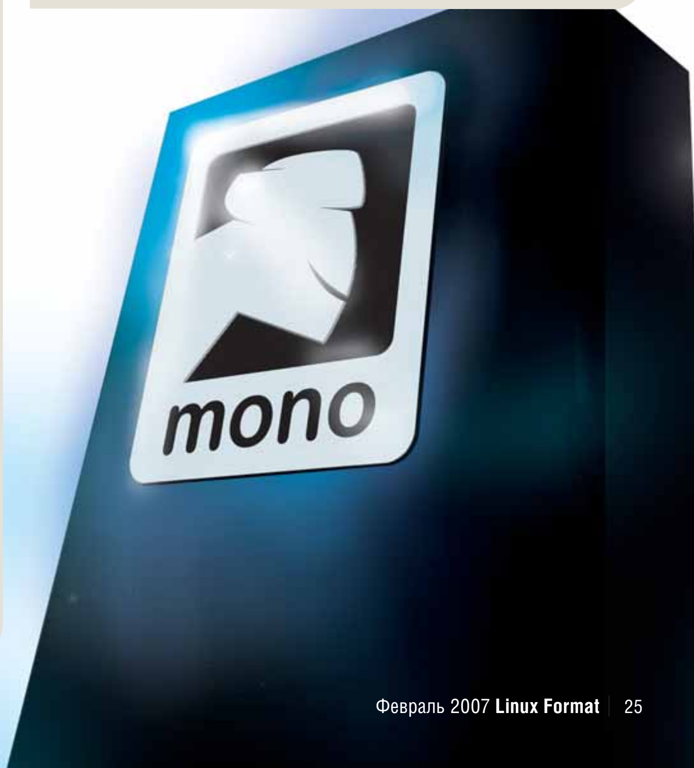
» http://f-spot.org Продвинутое средство организации и редактирования фотографий.

» www.beatniksoftware.com/tomboy Создание заметок стало проще.

» www.cmartin.tk/blam.html RSS-клиент *Blam*.

» www.ifolder.com Домашняя страница проекта *iFolder* – инструмента синхронизации файлов от Novell.

»



Счастье хакера

Управление памятью, быстрая разработка, простое распространение и многое другое.



Майкл считает...

Майклу Миксу нравится C#, но не ожидайте портирования *OpenOffice.org* на C# в ближайшем будущем – «переписывание всего с лету не делается», говорит он.

За любой хорошей программой стоит столь же хорошая команда разработчиков, и фактически, большинство проектов зарождаются потому, что разработчик нуждается в программе для решения определенной задачи, но не может найти существующее решение. Вы уже видели, как Mono используется для построения крутых приложений, но почему их разработчики выбрали в первую очередь Mono?

Основные преимущества Mono следующие:

» Его основной язык – C#, новый язык, изобретенный Microsoft. Он во многом основывается на Java и C++, но не включает в себя некоторые сложнейшие части C++, например, макросы и множественное наследование.

» Программы можно запускать в «управляемом» (managed) режиме, то есть Mono следит, помимо прочего, за выделением памяти. Это спасает программистов от необходимости выискивать утечки памяти: обо всем позаботится Mono.

» Мощная интегрированная среда разработки, называемая *MonoDevelop*, способна создавать графический интерфейс простым перетаскиванием виджетов.

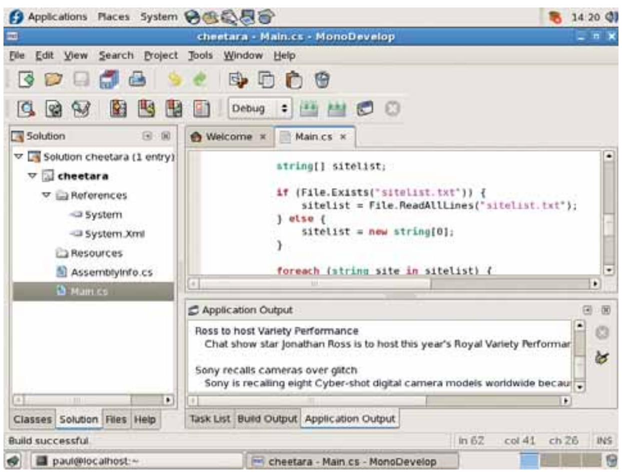
» Тесная интеграция библиотек Linux, включая *GTK#* для создания «родных» пользовательских интерфейсов для Gnome.

Все это значит, что программисты могут тратить меньше времени на выполнение черной работы и больше времени посвящать решению реальных проблем.

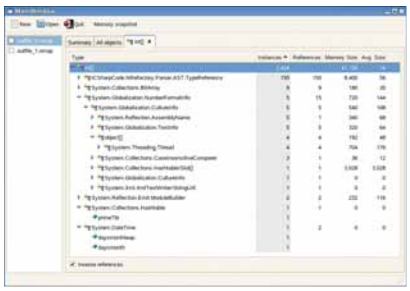
C# против C и Python

Большая часть рабочего стола Gnome написана на C, но уже несколько лет идут разговоры о том, можно ли использовать более новый язык высокого уровня – то есть один из тех, на которых чуть проще программировать и которые снижают время разработки. Сейчас, когда *GTK#* является официальной привязкой Gnome, программы, написанные на C#, можно поставлять как модули в Gnome. Это изменение имеет место в Gnome 2.16, с *Tomboy* – первым приложением, получившим официальное благословение.

Mono завоевывает множество поклонников. Джо Шоу из *Beagle* много экспериментировал как с C, так и с Python, и обнаружил, что C# превосходит любой из них. «За годы жизни я написал кучу кода и C, и на Python, и ни один из них не годится для больших приложений. C# занимает прекрасную нишу между ними: он воистину объектно-



› *MonoDevelop* – предпочтительная среда разработки для кодеров, работающих с Mono над большими проектами.



› *Heap-shot* делает снимок использования памяти вашей программой, и вы точно знаете, какими ресурсами пользуетесь.

ориентированный, четко типизирован, имеет проверку типов на этапе компиляции и способен работать в виртуальной машине с рефлексией и сборкой мусора», говорит он. «В языке C слишком много времени уходит на шаблонный код. Вы все время этим занимаетесь, и все время кажется, что есть способы и получше. Python – другая крайность. Он очень удобен для приложений-прототипов, и особенности этого функционального языка нельзя превзойти, но отсутствие статических типов и строгих интерфейсов очень усложняют создание повторно используемых компонентов с заданными свойствами. Большие программы на Python довольно быстро теряют сопровождаемость».

Шоу не единственный, кто так думает – Аарон Боковер считает C# выбором «без раздумий»: «Фактически здесь нет проблем совместимости, о которых нужно было бы беспокоиться. Никаких проблем с архитектурой, платформами, дистрибутивами, да и просто сборка исходных текстов на C# является намного более простой в управлении, чем в случае C. Язык C, конечно, имеет свою нишу, но в наши дни для пользовательских приложений разумнее использовать Mono».

Даже Майкл Микс – выдающийся хакер *OpenOffice.org* и программист со стажем на C/C++ – думает, что у Mono и C# есть преимущества. «В общем и целом, в долгосрочной перспективе я поставил бы на Mono, а не на C++ – по многим причинам: библиотеки «для всего» являются частью платформы, сборка мусора, увеличенная производительность и, конечно же, более чистый язык C#. Mono позволяет вам оставаться на знакомых языках: Python, C, C++, VB.NET и так далее, плюс вы можете повторно использовать множество стандартных библиотек и кросс-платформенных приложений в одной связке – лучшее из множества миров».

Спасибо за память

Хотя предпочтительнее всего писать в Mono управляемый код, где Mono полностью отслеживает использование памяти наиболее эффективным образом, C# позволяет также писать «неуправляемый» код, то есть делать эту работу самому. Конечно, при взаимодействии с «родными» библиотеками C другого выбора нет, а это означает проблемы, поскольку Mono не знает, как «родные» библиотеки используют память – за всем этим нужно следить вам. «Значит, что Mono временами будет менее расторопно прибирать за собой и это приведет к чрезмерному расходу памяти», говорит Ларри Юинг.

Джо Шоу из *Beagle* признается, что самой большой проблемой его команды было использование памяти. «Эти инструменты были слабы в Mono, но мы видим, что в сообществе и разработчики Mono, и разработчики приложений начинают обращать на это внимание. Кроме собственного профилировщика Mono, у нас есть профилировщики *Heap-buddy* и *Heap-shot*, позволяющие снизить потребление памяти. Мы поняли, что наделали глупостей в *Beagle*, нерационально обращаясь с памятью».

Heap-buddy и его новый компонент *Heap-shot* проектировались для предоставления точных отчетов об использовании памяти, помогаю-

щих разработчикам определять узкие места и утечку ресурсов. Они уже применяются и достигли значительного эффекта – сообщая о *Heap-shot* в своем блоге, де Икаса писал: «На прошлой неделе я снизил с его помощью потребление памяти в *Tomboy* на 340к», – разве плохо?

С *MonoDevelop*, Mono имеет собственную среду разработки, как любая серьезная программная платформа. *MonoDevelop* был портом Windows-приложения SharpDevelop, в котором хакеры заменили графический интерфейс *Microsoft Windows Forms* на *GTK#*. Сегодня *MonoDevelop* имеет функции управления проектами, подсветку синтаксиса, автодополнение кода и даже систему визуального проектирования графического интерфейса, известную как *Stetic*. Это намного быстрее, чем в прежние дни, когда вы набивали все элементы своего интерфейса как код.

Молниеносная разработка

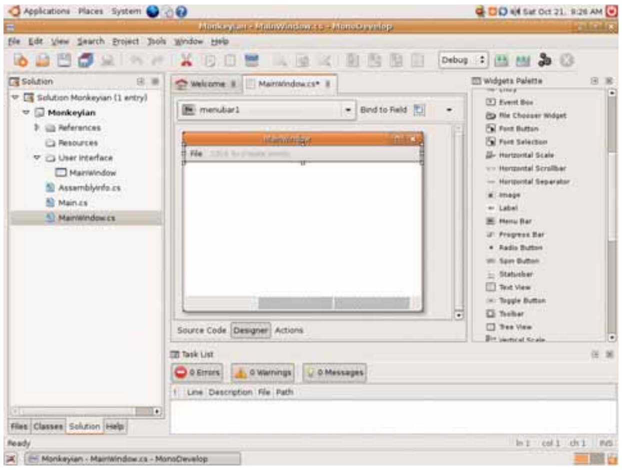
По своему назначению *Stetic* очень похож на *Glade*: он сохраняет графический интерфейс пользователя в XML, загружемый во время выполнения и транслируемый в виджеты *GTK#*. Однако *Stetic* более тесно интегрирован со средой времени выполнения Mono, а значит, способен реализовать функции, которые нельзя было бы сделать в не учитывающем особенности платформ дизайнере типа *Glade*. Луис Санчес [Luis Sanchez], ведущий разработчик *MonoDevelop* и ответственный за *Stetic*, рассказал нам: «*Stetic* имеет несколько преимуществ перед *Glade*: например, поддерживает произвольные виджеты и лучше интегрирован в Mono и *MonoDevelop*. С другой стороны, *Glade* – более зрелая технология, и дизайнер интерфейса *Glade* шире распространен. Так что это дело выбора. *Stetic* уже может читать файлы *Glade*, так что миграция с *Glade* на *Stetic* должна быть простой – *Glade* и *Stetic* вполне могут сосуществовать».

MonoDevelop приближается к релизу 1.0, и хотя основной набор функций считается завершенным, по ходу добавляется множество мелочей. Это включает поддержку автодополнения для параметров, импорт проектов *Microsoft Visual Studio*, *CVS* и *Subversion*, плюс улучшение интеграции с семейством утилит Autotools. Первый бета-выпуск *MonoDevelop* 1.0 должен был выйти к моменту прочтения этих строк, а финальный релиз пока нацелен на апрель. Как говорит Санчес, «основная цель этого первого стабильного релиза заключается в разработке приложений *GTK#*. *MonoDevelop* имеет почти всю требуемую для этого функциональность, и мы не собираемся добавлять много новых функций, нам просто нужно завершить отсутствующие части и исправить имеющиеся ошибки».

Теперь, когда *Stetic* является предпочтительным дизайнером графического интерфейса *GTK#* для Mono, некоторых интересует, будут ли ранние программы Mono – *Beagle*, *F-Spot*, *Blam* и другие – придерживаться *Glade* или мигрируют на *Stetic*, как только выйдет финальная версия *MonoDevelop* 1.0. Ларри Юинг пока не уверен: «Переход на *Stetic* сейчас означал бы массу работы, практически не видной пользователю, так что мы вряд ли будем спешить с изменениями», говорит он. «С другой стороны, новые функции в *MonoDevelop* и *Stetic* весьма заманчивы. Иногда, чтобы начать большие изменения, нужно только сильное искушение».

Платформы, поддерживаемые Mono

s390	Linux
SPARC	Linux, Solaris
PowerPC	Linux, OS X
x86	FreeBSD, Linux, NetBSD, OpenBSD, OS X, Solaris, Windows
x86-64	Linux
IA64 Itanium2	Linux
ARM*:	Linux
Alpha	Linux
MIPS	Linux



› **Накидайте виджеты мышью:** ***MonoDevelop*** **включает дизайнер интерфейса Stetic, начиная с версии 0.10.**

На плечах гигантов

Неважно, насколько мил язык программирования C# или сколько функций имеет *MonoDevelop*; успех Mono во многом определяется тем, насколько доступна из него обширная коллекция библиотек, уже имеющихся в Linux. К счастью, Mono упрощает сборку на основе библиотек C, так что можно разрабатывать полностью управляемые библиотеки Mono, которые затем легко разделяются между программами. Это

> «Некоторых интересует, мигрируют ли ранние программы Mono на Stetic.»

означает, что нечто вроде Boo – популярного Python-подобного .NET-языка, который мы рассматривали в LXF77 – можно встроить внутрь множества приложений, чтобы обеспечить поддержку сценариев.

Наученная предыдущим горьким опытом ада DLL, Microsoft придумала для .NET так называемый глобальный кэш сборок (GAC – Global Assembly Cache). Библиотеки, разделяемые множеством приложений, устанавливаются в GAC с учетом номера версии, то есть одна библиотека может быть в нескольких версиях одновременно. Именно сюда устанавливается большая часть ПО Mono, хотя некоторые разработчики могут выбрать использование локальных копий библиотек.

Локальная установка библиотек – это путь, выбранный *Banshee*, как поясняет Аарон Боковер: «В *Banshee* включены внешние исходные тексты и библиотеки (D-BUS, HAL, TagLib#, Mono.Zeroconf и Boo). Когда *Banshee* установлен, эти управляемые библиотеки инсталлируются как его приватные копии. Это помогает при работе с «нестабильными» библиотеками (их API может меняться в будущем, так что мы не хотим помещать их в глобальный кэш сборок), поскольку дает гарантию, что обновление какой-нибудь внешней библиотеки не испортит наше приложение. Это также обеспечивает очень гибкий контроль за разработкой».

Именно комбинация всех этих функций придает Mono остроту в случае с новыми приложениями – его легко изучать, легко использовать для быстрого кодирования приложений, и на десятках платформ он работает с невероятной скоростью. Стоит ли удивляться, что он взял штурмом программистский мир?

Уже грядет: .NET 3.0

Мы раскрываем планы на Mono 2.0 и дальше.



Мигель считает...

«Хорошая новость о C# 3.0 – хотя он выглядит гигантом в своих достижениях, но изменения в этом языке, вероятно, самые минимальные, которые когда-либо вносила Microsoft».

Если эта статья сделала свое дело, перспективы дальнейшей разработки Mono уже встопорщили волосы у вас на загривке. Но не расстраивайтесь, что мы начнем экскурсию в увлекательное будущее с остановки в Редмонде. Да, после долгой разработки, начатой в мае 2001 г., Windows Vista вышла и уже используется первопроходцами. Кроме новой внешности, Aero, есть кое-что поинтереснее: .NET 3.0.

Это следующее поколение платформы .NET, и Microsoft намеревается распространить ее на Windows XP и Windows Server 2003, охватив широкую пользовательскую базу. Но, в отличие от .NET 2.0, .NET 3.0 не меняет радикально нижележащую среду исполнения .NET. Вместо этого добавляются новые библиотеки для обработки графики (*Avalon*), коммуникаций (*Indigo*) и других задач. Также вводится C# 3.0, обладающий новыми функциями, например, лямбда-выражениями и анонимными типами.

Важно, что все это по-прежнему работает со старыми библиотеками .NET, и Mono не потребует изменений для работы с программами .NET 3.0. Mono уже имеет собственную реализацию .NET 2.0, дающую людям все преимущества .NET без чувства вины. Эти новые API вообще не меняют значение Mono, и фактически можно счастливо проигнорировать дополнения .NET 3.0 и по-прежнему цвести.

Не забуду .NET 2.0...

Разработчики Windows обдумывают преимущества .NET 3.0, и Mono тоже не стоит на месте. Идет работа по следующему крупному релизу, Mono 2.0, который должен выйти во втором квартале 2007 г. Как мы уже сказали, сейчас Mono поддерживает все из .NET 1.0 и 1.1, но лишь некоторые вещи из .NET 2.0. Сюда и направлены основные усилия: Mono 2.0 будет первым выпуском с полной поддержкой .NET 2.0. Он также будет включать новую систему сборки мусора, предотвращающую фрагментацию памяти при длительной работе программ и новый компилятор Visual Basic.NET, плюс анонсируется предварительная поддержка *Windows Forms 2.0*.

В дальней перспективе проект Olive попытается перенести часть самых интересных библиотек .NET 3.0 в фонд Mono. Сейчас работа во многом сфокусирована на Workflow API, что позволит разработчикам производить легко автоматизируемые приложения, и поддержке XAML: это *Glade*-подобный XML-язык, упрощающий создание кода графических интерфейсов. В планах на сегодня – сделать первые «беты» Olive доступными с Mono 2.2, наряду с финальной поддержкой *Windows Forms 2.0*, в конце этого года.

Пока ядро Mono движется к Mono 2.0, другие исследуют способы помещения имеющегося кода Mono для работы в других местах. Три проекта особенно впечатляют: *Grasshopper* от Mainsoft, Mono в *GCC* и *mkbundle*.

› Филипп Коэн (справа) вкалывает, чтобы разработчики .NET могли запускать свои приложения на Java – благодаря Mono.





› Стек .NET 3.0 добавляет *Avalon*, *Indigo* и другие новые компоненты в стек .NET 2.0.

Глубокая интеграция

Grasshopper – умная программа, выполняющая кросс-компиляцию кода .NET в код Java. Практически, это означает, что разработчик может писать на C# как обычно, а затем запустить эту программу на том сервере J2EE, на каком захочет. Mono занимает центральную место во внутренней работе *Grasshopper*, потому что *Grasshopper* должен конвертировать все библиотеки Mono в J2EE, чтобы основанные на нем программы могли работать в Java. Сейчас упор сделан на Windows-разработчиков, желающих освободиться от Windows, но Mainsoft – это также и крупнейший соавтор Mono после Novell, так что преимущества от этой работы получат все. Филипп Коэн [Philippe Cohen], вице-президент по разработке продукта в Mainsoft, сказал: «Впервые разработчики могут портировать все свои .NET-приложения в Linux, как клиентские, так и серверные, благодаря полной реализации *Windows Forms*, которая придет с этим релизом. Версия 1.2 – это также значительная веха на пути к поддержке .NET Framework 2.0. Мы надеемся на полный выпуск 2.0 в 2007 году».

Работа в *GCC* нацелена на производство работоспособного выходного интерфейса (back-end) к *GCC* для универсального промежуточного языка .NET. Сейчас *GCC* имеет множество входных и выходных интерфейсов. Код компилируется через определенный входной интерфейс. Например, *GCC* уже может читать C и C++, но также может читать Fortran, Java и другие. Затем код преобразуется в собственный промежуточный язык *GCC*, называемый Generic. Наконец, эта не зависящая от языка часть должна быть оптимизирована и пропущена через подходящий выходной интерфейс для текущей машины, где она конвертируется в машинный код.



› Студент Альп Токер хочет писать расширения для Compiz на Mono. Google помогает ему это сделать.

Причина, почему выходной интерфейс *GCC* для Mono очень важен – это означает, что код будет прочитан, конвертирован в Generic, затем отправлен в интерфейс Mono, и сможет запускаться поверх .NET и Mono. Преимущество здесь в том, что Mono неожиданно открывается для любых языков, поддерживаемых *GCC*, исключая необходимость для команды Mono писать собственный компилятор для этих языков.

Наконец, *mkbundle* – это программа, спроектированная для того, чтобы позволить приложениям Mono работать в системах без установленного Mono. Сказанное выглядит невозможным, и сделать это – непростая задача, но *mkbundle* работает путем статической компиляции программ Mono, создавая, по сути, исполняемый файл Linux, который содержит встроенные версии Mono наряду со всеми используемыми библиотеками. Преимущества очевидны: программы больше не требуют сложной процедуры установки; вы можете просто щелкнуть по файлу, и он запустится. Недостатком метода, конечно, является создание на редкость громоздких программ, не разделяющих библиотеки с другим ПО.

Mono в Linux

Мы уже видели, как несколько проектов Mono нашли свой путь в большинство популярных дистрибутивов (*Tomboy*, *F-Spot* и *Beagle* доступны как стандарт в трех самых популярных), и над какими будущими функциями работают их авторы. Но на горизонте есть и другие замечательные вещи, например, проект Mono.Fuse. Fuse – это модуль ядра, позволяющий файловым системам работать в пространстве пользователя, то есть непривилегированные пользователи могут создавать собственные файловые системы без необходимости пересобирать ядро или иначе обходить ограничения системы.

Mono.Fuse расширяет модуль Fuse на Mono, позволяя программистам собирать «обертки» (wrappers) к файловым системам, используя управляемый код на C# или другом .NET-языке на выбор. Чтобы вы поняли его потенциал, Fuse уже используется многими файловыми системами, включая EncFS (она шифрует данные на лету!), Wayback (контролирует версии файлов!), TagFs (позволяет просматривать свою коллекцию музыки по ID3-тегам!), и Flickrfs и Wikipediafs (угадайте сами, что они делают!).

Вся соль в том, что Fuse уже чрезвычайно популярен – несмотря на то, что вынуждает людей использовать библиотеки и API языка C. Связка с Mono откроет поле для работы гораздо большего числа программистов, потому что C# легче в использовании, и кроме того, поставляется с весомой коллекцией библиотек .NET.

Еще одно отражение идеи взять крутую технологию и сделать ее более доступной – проект, выполненный в рамках акции Google Summer of Code, Mono/*Xgl*, за который студент Альп Токер (Alp Toker) взялся под наставничеством ответственного за Mono Массимилиано Мантионе [Massimiliano Mantione]. Цель здесь заключается в такой модификации вундеркинда графики *Compiz*, чтобы программисты могли писать к нему расширения, используя Mono и C#. Когда вы прибавите сюда мощь каркаса Тао, описанного на стр. 25, делать крутейшие трехмерные эффекты всего несколькими строками кода окажется неожиданно легко.

Ссылки для разработчиков

» www.mono-project.com Скачайте последнюю версию Mono для вашей платформы и испытайте ее.

» www.monodevelop.com Домашняя страница MonoDevelop, включающая дизайнер графического интерфейса Stetic в качестве стандарта.

» www.mainsoft.com Если у вас есть друзья, прикованные к Windows, Mainsoft может помочь им портировать свои приложения на Java.

» www.jprt.com/Blog/archive/development/mono/2006/Aug-29.html Узнайте больше о Mono.Fuse на этой странице, но будьте осторожны: она все еще перерабатывается!

» www.taoframework.com Каркас Тао для разработки игр под .NET.

» http://unity3d.com Домашняя страница закрытого движка Unity 3D.

» www.msdn.com Официальная документация Microsoft для .NET и C# полностью применима и к Mono. Mono имеет и собственную документацию, но, по словам де Икасы, «это та область, где я черпаю информацию с большим эффектом», так что попробуйте тоже.

> «Fuse уже популярен: добавка к нему Mono откроет поле для множества программистов.»

Ваш следующий шаг...

Теперь, когда Gnome распахнул двери, чтобы принять больше программ Mono, появилась надежда, что мы увидим шквал новых разработок, когда люди поймут преимущества новых возможностей. Мы прошли точку тревоги за проблемы с патентами, и впереди Mono ждет прекрасное будущее. Что будет в следующем *Beagle* или *F-Spot*? Как использовать все разнообразие библиотек для приближения Linux к принятию на рабочем столе?

Здесь нет простых проблем, но вы можете помочь, даже если вы не программист – потому что мы объявляем конкурс «Сделай это с Mono!», который даст вам шанс увидеть приложение своей мечты, написанное для Linux – см. врезку «Сделай это с Mono!» ниже, чтобы узнать, как. LXF



Сделай это с Mono!



Расскажите нам о приложении своей мечты, и мы, возможно, даже сделаем его для вас!

Нет ничего хуже, чем иметь великолепную идею программы, которую вы хотели бы видеть в Linux, но быть не в силах написать ее самому. Поэтому Linux Format объединился с Novell и организовал конкурс «Сделай это с Mono!»: сообщайте нам свои идеи самого сногсшибательного приложения Linux. Все заявки будут опубликованы в сети, и каждый сможет затем проголосовать за приложение, которое сочтет лучшим.

Заявка, получившая наибольшее число голосов на дату закрытия (2 апреля 2007 года), будет выполнена на Mono и сделана доступной под GPL для использования всеми желающими. Авторы 30 лучших идей выиграют футболки Mono, так что не будет беды, если вы тоже попробуете!

Photography: Jason Kaplan

Непохожий воитель

Майкл Тиманн вошел в мир свободного ПО благодаря своей выдающейся работе над компилятором GNU, но главный его вклад в этот мир – и вклад не единовременный, но постоянный – это его деловой ум. Сегодня он обсуждает Red Hat, доходы и «*Искусство войны*» с *Linux Format*.

Немного найдется пользователей Linux, которые не слышали бы о *Cygnus* Solutions [*Cygnus* в переводе – созвездие Лебедя, в нем, между прочим, находится туманность Северная Америка – емкое название! – прим. ред.]. Эта фирма известна поддержкой немалой части основных программ GNU, и она значительно облегчает пользователям Microsoft Windows знакомство с возможностями Unix через свою библиотеку Cygwin. Майкл Тиманн [Michael Tiemann] был одним из соучредителей *Cygnus*, равняющихся на Манифест GNU, который он считает просто замаскированным бизнес-планом. Когда *Cygnus* Solutions слились с Red Hat в 1999, Тиманн стал техническим директором. Именно эта роль и обусловила его вклад в развитие Open Source, приведя его на пост, который он занимает сейчас – вице-президент Red Hat по вопросам Open Source. Грэм Моррисон недавно имел возможность пообщаться с Майклом и попытался выудить некие воспоминания и мысли из глубин сознания одного из самых интересных и влиятельных людей в мире свободного программного обеспечения.

Linux Format: Что означает должность вице-президента по вопросам Open Source?

Майкл Тиманн: Подразумевается, что я должен общаться с теми, кто определяет политику и стратегию [использования] Open Source, и в частном, и в общественном секторе. Сейчас я готовлюсь к поездке на Шри-Ланку и в Индию, где мы должны побеседовать с представителями правительства и частного сектора об их выросшей осведомленности и об использовании открытых решений. В июне я был в Токио и Осаке, и встречался с людьми, которые определяют политику в сфере образования, правительственную политику, материально-техническое снабжение и, конечно же, инфраструктуру ИТ.

LXF: Значит, именно с вами нужно общаться, если хочешь узнать, как продвигать Open Source?

МТ: Я думаю, скорее, если хочешь понять, что означает Open Source на уровне стратегии. Вы, наверное, знаете, что я создал первую в мире компанию открытого ПО, в принципе, первую фирму свободного ПО, еще в 1989…

LXF: Cygnus Solutions.

МТ: Да. И некоторое время я занимался этим. Я видел множество успехов, несколько провалов, немало рационализаций и реализаций, поэтому главное, что я должен делать – это соединять все точки, и следить за этим соединением.

LXF: Что привело вас к созданию *Cygnus*?

МТ: В 1987 году я начал работать над компилятором GNU C++. Точнее, над компилятором GNU C – компилятора C++ тогда еще не было. И привел меня к этому мой всегдашний интерес к компиляторам. Когда я учился в школе, мне казалось просто волшебством, что можно взять высокоуровневый язык и превратить его в нечто исполняемое машиной. И то, что делал потом, было… Мне просто чудом казалось, что я могу получить исходный код компилятора совершенно бесплатно! »





O RED HAT

«Мы определили бизнес-модель Open Source.»

» **LXF: Ах, вот в чем главная причина вашего присоединения к миру свободного ПО?**
MT: Да, это было большим плюсом. Представляете, что было бы, если бы Кит Ричардс еще мальчишкой зашел в магазин музыкальных инструментов и выбрал себе гитару бесплатно? Это такое же чудо. Для меня это было просто невероятно.

Самая первая идея пришла мне в голову, когда я работал на компьютере с чипом National Semiconductor 32032 – компилятор GNU C его не поддерживал. Он поддерживал рабочие станции Sun – а у меня ее не было. Он поддерживал VAX – и VAX у меня тоже не было. Но он поддерживал многое другое, и я подумал: «А не попробовать ли портировать этот компилятор, чтобы он работал как родной на доступной мне машине?»

В то время работа по созданию порта компилятора выполнялась лишь несколькими малоизвестными компаниями, и они бы выкатили счет в несколько миллионов долларов и потратили несколько лет на попытку его сделать. Я уселся за эту задачу, и через пару недель отправил сообщение по спискам рассылки, в котором говорилось: «Я сделал его, и он генерирует код, который на 20% быстрее, чем родной компилятор, поставляемый с этой машиной».

LXF: И что же вы исправили?
MT: [Смеется] Видите ли, эти компиляторные компании – они тогда были такими мелкими лавочками. Не знаю – я на самом деле, в самом прямом смысле, не знаю, почему они делали все так долго и брали так дорого. Но я сказал себе: «Эй, если эта теория свободной рыночной экономики – это стоящая теория, то моя мышеловка лучше!» Правильно? «Если я могу делать что-то, что работает настолько быстрее и настолько лучше, да еще настолько легко – ведь это может стать бизнесом».

Я не хотел становиться бизнесменом, я хотел быть программистом. И я потратил два года на то, чтобы уговорить ребят с предпринимательской жилкой войти в этот бизнес со мной. Никто не хотел этого делать, потому что никто не мог понять, как можно делать деньги на свободном программном обеспечении.

LXF: Вы явно были мастером своего дела. Вы могли бы запросто создать фирму, делая то же, что и все, но вдвое быстрее, и оставить ее коммерческой – почему вы этого не сделали?
MT: А вы рассмотрите обе стороны. Одна сторона – за две недели я сделал то, на что должно было уйти два года. Другая сторона – мне очень помогли своими дельными предложениями другие люди, они сделали мою работу вдвое эффективнее и лучше. Зачем же мне было этого лишаться? В то время – да, собственно, и сейчас – меня куда больше интересовали Возможности, с большой буквы, нежели просто возможность сделать что-то лучше других и прикарманить их деньги.

О НАЧАЛЕ

«Бизнесменом я быть не хотел, я хотел быть программером.»

LXF: А что было после компилятора?
MT: Через два года меня озарило: ведь если все кругом говорят, что идея отличная, но никто не верит, что она работает, то у меня не будет конкурентов. Я вычислил, что технические преимущества и отсутствие конкуренции обеспечивают беспроигрышный вариант, вот все это вместе и вдохновило меня затеять *Cygnus*. Я нашел еще пару человек…



LXF: Значит, вы все-таки получили поддержку?
MT: Я нашел пару человек с такими же убеждениями, которые хотели дать этому делу толчок. Одним из них был Джон Гилмор [John Gilmore], еще один – Дэвид Хенкель-Уоллес [David Henkel-Wallace]. До *Cygnus* Джон славился как работник номер пять в Sun. Он трудился на нелегкой ниве портирования большой части BSD… Билл Джой [Bill Joy, автор BSD и соучредитель Sun], тот в Berkeley развлекался, а Джон был одним из тех, кто портировал его работу в платформу Sun.

LXF: И как вам удалось убедить их, что это действительно бизнес-модель?
MT: Обычно я приходил в фирмы, где использовались компиляторы, и говорил: «Вот что мы предлагаем. Выгода вам понятна?» Одной из первых фирм, увидевших тут большую выгоду, была Intel. Стив Мак-Гиди [Steve McGeady] возглавлял проект по разработке микропроцессора i960, очень важного микропроцессора для бизнеса телефонии Intel – среди их клиентов были фирмы типа Nortel, которая создала солидные продукты в сфере телефонии и передачи данных. Мак-Гиди сказал: «Мне нужен хороший компилятор для моих разработчиков, такой, за который им не придется платить. Если они согласятся оплачивать вам услуги поддержки, договаривайтесь с ними сами, а я хочу сделать наш чип популярным, удешевив инструменты разработки». Ну, мы и сказали: «Отлично: платите один раз, потом распространяйте, а если кто-то из последующих клиентов захочет иметь с нами бизнес – пожалуйста!»

LXF: Значит, Intel заплатил за исходное оборудование…
MT: Верно, они оплатили так называемую разовую инженерную разработку [non-recurring engineering, NRE], причем заплатили нам не только вдесятеро меньше цены проприетарных подрядчиков, но и вчетверо меньше того, во что им обошлась бы собственная разработка. Мы предложили им крайне дешевую и эффективную сделку, но вскинулись-то они по другой причине. Их взбудоражила возможность изменить динамику рынка, и в результате Intel создал успешнейший бизнес на рынке встраиваемых технологий. А потом к нам пришли другие фирмы – NEC, Hitachi, Fujitsu, Motorola – да я даже всех названий не помню, но всем этим ребятам, занимавшимся встраиваемыми технологиями, ужасно хотелось предлагать весьма конкурентоспособные инструменты разработки.

В итоге набор инструментов для компиляции GNU стал своего рода стандартом бизнеса встраиваемых технологий. Мы буквально прошли путь от странной «нишевой» бизнес-модели до определяющей технологии в индустрии встраиваемого ПО.

LXF: А велик ли был Cygnus в то время?
MT: Мы создали фирму с доходом свыше $20 миллионов. Когда Red Hat купил *Cygnus* [в начале 2000] у нас было примерно столько же работников и такой же уровень дохода, что и у Red Hat. У них была очень удачная первичная эмиссия [акций]. Для нас стало проблемой то, что на финансовом рынке операционные системы ценятся намного выше компиляторов. Как пляж против пахотной земли!

LXF: Я всегда считал это слиянием.
MT: Ну, можно и так считать… Если говорить буквально, мы обменялись акциями, и к концу дня мы все стали акционерами Red Hat. Но бизнес *Cygnus* был очень сильным, многие из *Cygnus* до сих пор работают в Red Hat, основные разработчики компилятора GNU, которые работают в Red Hat – и отлично работают – наши. Я хотел отметить еще один момент, но как-то позабыл…

LXF: А как в Cygnus пришли к Windows?
MT: Платформа Windows была очень важной для разработчиков встраиваемых технологий. Им нравились инструменты GNU, но они хотели использовать дешевые ПК. Мысль о затрате $20 000 на проприетарную рабочую станцию…

LXF: Это было еще до Linux?
MT: Об этом-то я и хотел сказать, спасибо! Вот вы думаете про себя: «Майкл, ну зачем ты сдуру занялся компиляторами вместо ОС?» А ответ в том, что AT&T как раз



› Тиманн работает также в правлении Embedded Linux Consortium и в технических советах Jabber и Gnome Foundation

выставило иск Berkeley [Berkeley Software Design, насчет ОС BSD], и все это смотрелось очень шатко с точки зрения закона, и было это за два года до рассылки знаменитого сообщения Линуса Торвальдса «Я делаю ОС». Ну не пришло нам в голову, что можно получать доход от несуществующей технологии! Существовал GNU – ну, ядра GNU еще не было, но компиляторы GNU, черт возьми, были, и мы создали свой бизнес на этой первой технологии.

LXF: Вас не соблазнила работа над Hurd?

MT: Не особенно. В Америке есть пословица: «Первопроходцам – стрелы, поселенцам – урожай». Слово «первопроходец» – не случайное в моей жизни [Red Hat называет Тиманна «подлинным первопроходцем ПО с открытым кодом»]!

LXF: Итак, версия для Windows…

MT: Да, надо было учитывать стоимость ПК, к тому же процессор Intel начал обгонять по производительности процессор Unix. Насмотревшись на парней с Windows-платформами, намного более быстрыми и дешевыми, чем дорогостоящие рабочие станции Unix, один из наших разработчиков заявил: «К черту это все. Создаю среду эмуляции Unix, будем работать с инструментами GNU на платформах Windows». Он создал библиотеку *Cygwin*, которая существует по сей день. За всю историю *Cygnus Cygwin* загружали в десять раз больше, чем любой другой наш продукт.

Это доказывает, что рынок Windows был крупным рынком, и даже сегодня, несмотря на все наши успехи в Red Hat, Windows остается крупным рынком. Но на моем компьютере Linux стоит с того самого дня, как я присоединился к Red Hat, и я в восторге от всего того, что там есть. Не вижу ни одной причины использовать Windows.

LXF: Как произошло слияние с Red Hat?

MT: С одной стороны, это произошло потому, что *Cygnus* был невероятно ценной собственностью. Мы были набором инструментов, которая определяла возможности и перспективы разработки приложений. А кому интересна ОС без приложений? То есть мы думали о том, «как это сделать»… *Cygnus* был первой фирмой, взявшей венчурный капитал в 1997 году: мы закрыли январь того года с Greylock, которая позже инвестировала в Red Hat, и August Capital, чем-то вроде заначки от Kleiner Perkins. Эти венчурные люди стали обдумывать стратегию ухода, рассмотрели создание IPO для компилятора – и подумали: «Не-а, это не получится». Но слияние показалось нам потенциально интересной идеей, и мы начали переговоры с разными дистрибутивами Linux. Все произошло стремительно. Большой интерес к нам проявили в Red Hat, и нам они понравились больше всего.

LXF: Какими вы видите перемены в Red Hat?

MT: Red Hat сместила свою бизнес модель с розничных продаж – ну, эти коробки на полках – в область поставок на предприятия. Мы определили бизнес-модель для Open Source. Конечно, мы не претендуем на единственность своего решения, но я не знаю ни одной фирмы с доходом в сто миллионов долларов, не говоря уж о нескольких сотнях миллионов, полученном исключительно от ПО с открытым исходным кодом и его обслуживания. В Red Hat мы изменили модель релизов, сместили наши акценты в бизнесе и поменяли клиентов, с которыми мы работаем. Некоторые до сих пор не могут простить нам утрату возможности купить продукт за $79 из распродажной корзины магазина дешевой электроники.

LXF: Задним числом понятно, что для Red Hat это стало безусловно выгодным решением, но в тот момент, что подвигло на сокращение коробочных версий Red Hat Linux и переход на корпоративный рынок?

MT: Если вы читаете книги о бизнесе, если вы читали книги Майкла Портера [Michael Porter] по конкурентной стратегии, так там говорится, что в принципе есть три конкурентных стратегии: вы можете производить продукцию или с низкой стоимостью, или с высокой ценностью, или для определенной ниши рынка. Он [Майкл Портер, – прим. пер.] говорит о так называемой стратегической дилемме: фирмы, пытающиеся выбрать более одной стратегии, неизбежно терпят крах, поскольку попытка соответствовать одной стратегии неизбежно мешает выполнению другой. Альберт Эйнштейн перефразировал эту дилемму, сказав, что нельзя одновременно готовиться к войне и…

LXF: …и пропагандировать мир.

MT: Совершенно верно. Все связанное с коробочным продуктом на корню губит любую попытку заняться бизнесом с предприятиями. Например, если вы хотите торговать коробочным продуктом, вам потребуются частые релизы. Вам не нужен продукт, который может пролежать на полке семь лет, лучше торговать таким, где каждые полгода появляется что-то новое. Посмотрите на доходы Red Hat, полученные тогда, когда они занимались коробочными дистрибутивами – вот как это выглядело: «Выпускаем новый релиз». «Зачем?» «Нужно повысить доход». А придите на предприятие: последнее, что им требуется, это бесконечные обновления. Они выбирают и технологию, и ее временные рамки, и больше всего ценят ту платформу, которая длительное время остается стабильной.

LXF: Странно, что никто другой до этого не додумался.

MT: Одна из истин, которой я научился за почти 20-летний опыт продаж ПО с открытым кодом, заключается в том, что нельзя победить по принципу «и я тоже так буду». Сунь Цзы в «Искусстве войны» говорит, что если хочешь атаковать тех, у кого сильна оборона, нужно иметь силу, вчетверо превосходящую силу обороняющегося – а лучше вдесятеро.

Вернемся к моей истории с компилятором GNU C: кто пошел бы на сделку с 23-летним юнцом? Тот, кто хочет иметь вдесятеро. Если я предлагаю производительность в четыре раза выше за одну десятую цены, это привлечет внимание на 20% быстрее, а «стоимость на 20% ниже» останется не у дел. Результат не всегда сногсшибательный, но, в общем, если какой-нибудь торговый агент предложит мне внушить кому-то, что экономия в 10% спасет их бизнес, я его выставлю вон и скажу, что больше этим не занимаюсь.

LXF: Сурово.

MT: Прелесть в том, что рынок открыл нам эту дверь, и мы утвердили свою конкурентоспособную позицию. Никто пока не придумал, как увеличить производительность в четыре раза за одну десятую стоимости Red Hat. LXF



Читайте больше!

Майкл считает стоимость ненадежных проприетарных программ и впадает в лирику, рассуждая о своем новом синтезаторе: на www.linuxformat.co.uk/tiemann.html.



ТОП-10: расширения Firefox

Общеизвестно, что *Firefox* умеет практически все – *Linux Format* решил разузнать у **Ричарда Коббетта**, до каких высот можно поднять версию 2.0.

После выпуска *Firefox 1.0 Mozilla* Foundation предприняла три на редкость хитроумных шага. Во-первых, правильно продвигала свой продукт на рынке, поясняя миллионам людей, почему *Internet Explorer* и подобным ему браузерам место на свалке. Во-вторых, постаралась заставить *Firefox* работать блестяще – ну, тут объяснять нечего. В-третьих, реализовала расширения. То, что раньше было прерогативой крупных программных фирм, неожиданно пошло по рукам надомников, азартно принявшихся расширять и улучшать базовый механизм. Появились и проекты-малютки вроде интегрированных читалок новостных лент, и проекты-гиганты типа *Songbird*, замещение *iTunes* на базе *Mozilla*.

› **Рекомендуем брать расширения для *Firefox* на сайте https://addons.mozilla.org.**



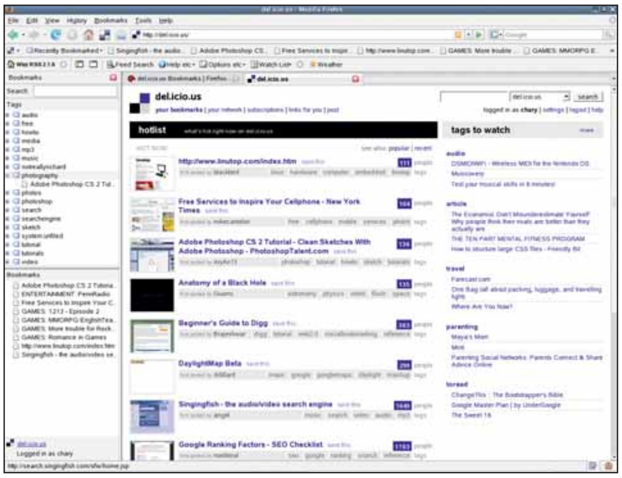
В последующие два года от концепции *Firefox* как сугубо базового браузера, допускающего расширение возможностей, стали отходить, и каждая новая версия норовит впихнуть в ядро все больше функций, ранее отдаваемых на сторону. Например, версия 2.0 отобрала у расширений восстановление сессии и проверку орфографии. Тем не менее, тысячи расширений по-прежнему доступны в репозитории *Mozilla*.

Какое из них предпочесть? Это зависит от вашего вкуса и рода занятий. Конечно, есть расширения, без которых просто немыслимо обойтись, но есть и такие, что диву даешься, зачем вообще понадобилось их разрабатывать. К последним можно отнести расширение Abe Vigoda Status Alert – оно отслеживает, жив еще или нет Эйб Вигода [американский актер, которого пресса почему-то не раз по ошибке хоронила, – прим. пер.]. При полнейшем равнодушии сетевой публики это расширение ликвидировали, и никто по нему особо не скучал.

Ограничившись обзором расширений первого типа, мы попробуем выбрать на странице https://addons.mozilla.org десять таких, которые наверняка понравятся читателям *Linux Format*. Если мы пропустили ваше любимое, то это не со зла – напишите нам, и в другой раз мы его не забудем. Начнем с двух расширений, относящихся непосредственно к браузеру: одно – популярное, другое – не очень.

Firefox без границ

1 Начнем с Greasemonkey: ни одно расширение еще не завоевывало мир с такой быстротой. Загрузите его с https://addons.mozilla.org/firefox/748, зайдите на сайт его подключаемых скриптов www.userscripts.org и полюбуйтесь, что оно умеет делать. Каждый скрипт – фрагмент кода на языке JavaScript, запускаемый авто-



› Управляйте закладками в Интернете с помощью Delicious Bookmarks, но не забывайте, что они публичны.

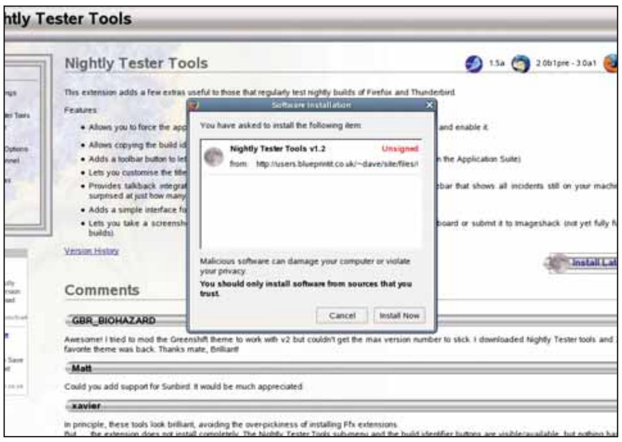
матически при входе на соответствующий сайт и способный на все, от радикального переформатирования страницы до наделения сайта новыми функциями. Например, любители сервиса Gmail смогут настроить его использование по ссылке **mailto:**, добавить к тексту письма HTML-подпись, зашифровать почту, присвоить меткам цвета, убрать с глаз долой папку со спамом… и так далее, и тому подобное. Полный список скриптов приведен на страничке http://userscripts.org/tag/gmail. Мы обнаружили 48 скриптов для Gmail, около 200 – для Flickr, 78 – для Delicious и 118 – для Yahoo. Чтобы разобраться с написанием скриптов, зайдите на сайт www.diveintogreasemonkey.org. Надо отметить, что Opera также предлагает подобные скрипты (фактически, начав это делать даже раньше), под маловыразительным названием UserJS, но ни в какое сравнение с Greasemonkey он не идет.

> «Можно зашифровать почту, убрать окно спама, вставить подпись… и так далее.»

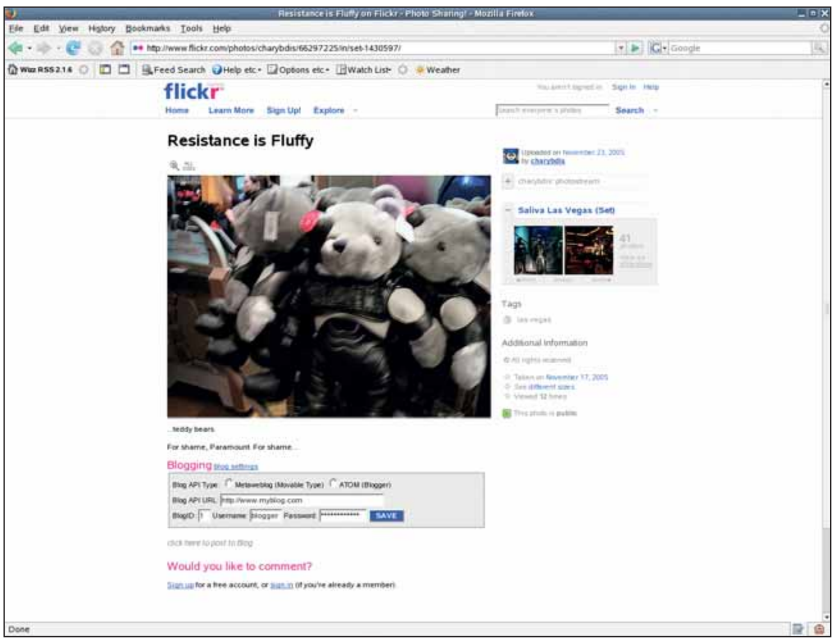
2 Следующий набор, Nightly Tester Tools, находится на http://users.blueprintit.co.uk/~dave/web/firefox/nightly.

При установке каждое расширение объявляет, какая версия браузера ему нужна, и в случае несовпадения, отказывается работать – например, расширения *Firefox 1.2* не запустятся в *Firefox 1.3*. Nightly Tester Tools позволяет пресечь эти разборки, разрешая запуск расширений, якобы несовместимых с текущей версией браузера. Поскольку немало расширений тихо отмирает, для пользователей это простейший способ разобраться, вправду ли произошло фатальное изменение или же они пали жертвой паранойи обновлений.

Nightly Tester Tools разрабатывался под Windows, но сейчас имеет полную поддержку для Linux и интегрируется с системой *Mozilla* »



› На вид Nightly Tester не очень красив, но очень скрашивает ожидание при обновлении расширений.



› Заметили, как Greasemonkey переделал эту страницу? Окно блога – новая опция поверх базового интерфейса Flickr.

Больше, чем Web

3 FTP и RSS обусловили появление немалого количества расширений *Firefox*, которые берут на себя черновую работу в Интернете. FTP через браузер – обычно тяжелая штука, но с FireFTP (https://addons.mozilla.org/firefox/684) вы можете получить приличное качество пересылки файлов откуда угодно. Но если вы любитель рискнуть или не имеете доступа к FTP, Gmail Space (https://addons.mozilla.org/firefox/1593) проделает примерно то же через пользовательскую запись Google Mail.



› С RSS от Sage, пульс всего мира на кончиках ваших пальцев.

4 Большинство расширений для чтения RSS обладают примерно одинаковыми возможностями, отличаясь разве что интерфейсом. Одно из старейших, прекрасно подходящее для пользователей, ценящих легкость – расширение Sage (http://sage.mozdev.org), оно отображает новости в формате газетной колонки.

Расширение NewsFox (https://addons.mozilla.org/firefox/629) хотя и не вошло в нашу десятку, но тоже неплохое: оно отображает RSS-ленту через привычный трехпанельный интерфейс в окне браузера. А The Wizz вызвал нашу симпатию своей скромной саморекомендацией: «Неплохая читалка RSS и Atom». Хоть он и не кичится своими достоинствами, но прекрасно работает с большим количеством лент и легко интегрируется с механизмом подачи новостей в *Firefox 2.0*.

Мы могли бы упомянуть еще о многом. Есть ведь расширения и для обучения слепой печати – Typing Tutor (https://addons.mozilla.org/firefox/2828), и для составления списка покупок – Grocery List Generator (https://addons.mozilla/org), и для многопользовательской игры Pong через сеть (https://addons.mozilla.org/firefox/554). Интернет – это не только web, и *Firefox* делает от него ответвления во все новые сферы.



› Web Developer Toolbar разбивает сайт на компоненты, и легко найти тот, что хромает.



› StumbleUpon – наркотик для любителей «растекаться мыслями по древу»: прыгать с ним по сайтам можно часами.

» Talkback, а что всего ценнее, может формировать отчет в случае ошибок в работе. Правда, Nightly Tester Tools не пытается решать за расширения те проблемы, что вызываются сменой версии, а просто позволяет им запуститься. Так что сами следите, как бы чего не вышло.

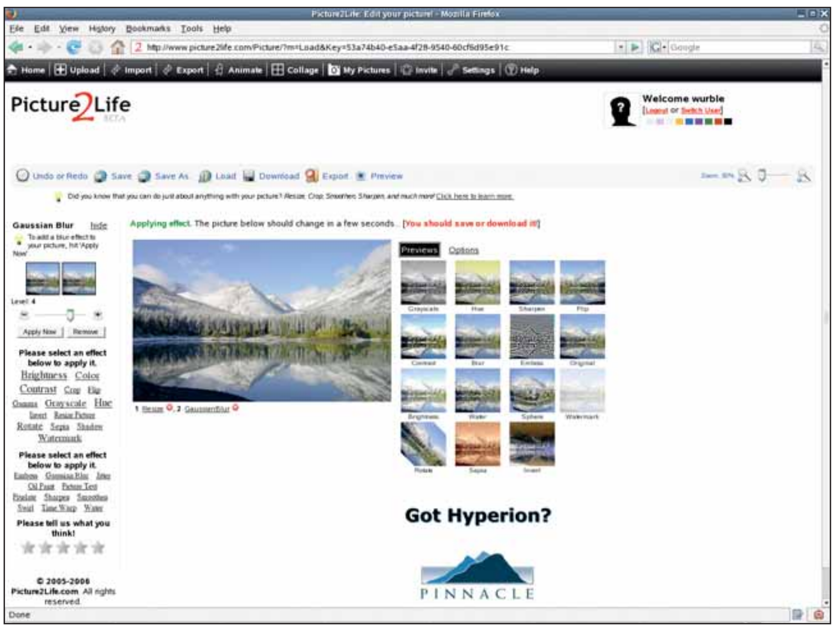
Закладки для всех

Закладки вроде бы простая вещь, но держать их под контролем на удивление сложно. Если вы не хотите потерять их при крахе системы, если вам нужны не просто ссылки на сайты или у вас больше одного компьютера, без расширений не обойтись. Мы выбрали два, с разными подходами.

5 Сейчас становится все популярнее держать закладки не у себя, а в сети, даже несмотря на сопутствующую угрозу безопасности. Для этого предлагается много сервисов, включая Chipmarks, Delicious и Google Notebook, и большинство из них работает через расширения *Firefox*. Но мы выбрали Delicious Bookmarks (с **https://addons.mozilla.org/firefox/3615**). Обычно меню Bookmarks быстро обрастает невероятным количеством ссылок, в которых невозможно разобраться, а Delicious Bookmarks распределяет их по категориям. Чтобы пользоваться расширением, нужно зарегистрироваться на сайте **http://del.icio.us**, там и будут храниться ваши закладки (помимо *Firefox*, еще одна причина им пользоваться – высокая активность сопутствующих проектов). Расширение официально одобрено Yahoo, а это увеличивает его шансы на долгожительство.

6 С другой стороны, для тех, кто ищет «то – не знаю что», есть StumbledUpon (**http://www.stumbleupon.com**), с почти четырьмя пустыми страницами. Это даже не новая панель инструментов, а наркотик! После его установки и регистрации на сайте, на панели инструментов вашего браузера появляется кнопка Stumble; нажав её, вы попадаете на web-страницу, выбранную случайным образом, но с учетом ваших интересов. Самое блестящее, что можно не просто указать тему, но и уточнить свой профиль, тут помогут кнопки Thumbs Up (Круто!) и Thumbs Down (Нафиг, нафиг...). Если сайт вам понравился, нажмите первую, и программа подберет вам кучу аналогичных; а если нет, жмите вторую, и поиск продолжится; при этом ваш профиль обрастает подробностями. Можно выставлять страницам оценки или (при желании) общаться с единомышленниками.

› Редактор изображений Picture2Life, запущенный на локальном расширении сервера POW.



Разработчикам

7 Понятно, что *Firefox* в первую очередь позаботился о web-разработчиках. Расширение Web Developer Toolbar (**https://addons.mozilla.org/firefox/60**) иметь просто необходимо. Его возможности неисчислимы: тут и отображение позиции и контура элементов страницы, и автоматическое масштабирование окна браузера для просмотра сайта в различных разрешениях экрана, и валидация HTML/CSS, и средства управления формами; чего только нет! Это отличный способ определить проблемы вашего сайта либо поглубже закопаться в другие. Если вы делаете сайты, без этого расширения и жизнь будет не мила.

8 Ищете что покруче? Попробуйте POW (Plain Old Webserver, **https://addons.Mozilla.org/firefox/3002**), локальный сервер для вашего браузера. Он пока существует в ранней версии 0.06, но служит отличным примером одаренности расширений *Firefox*. Он основан на JavaScript и поддерживает протокол HTTP Post, cookies, SQL и Ajax. Свой сайт с файлами вы можете защитить паролем, которых хранится в **/home/USER/.Mozilla/firefox/default.NUMBER/pow/htdocs**. Интересно, что уже разработаны расширения для POW. Одно из них,

Выбери меня

Расширения *Firefox* – труд легионов разработчиков, приковавших себя к компьютерам, чтобы сделать Сеть чуточку комфортнее, и они не собираются складывать руки. Когда мы выбирали первую десятку расширений, мы не руководствовались ни отзывами о них, ни даже их возможностями. Насущно необходимое для одного может быть источником досады для другого. Мы просто хотели возможно полнее отразить многообразие достижений и тот потенциал, который делает *Firefox* лучшим из браузеров. А что в вашей первой десятке? Расскажите нам о своих фаворитах: **letters@linuxformat.ru**.

Breadcrumbs (**https://addons.mozilla.org/firefox/2954**), сохраняет посещенные страницы и встраивает в *Firefox* систему для их поиска на будущее.

Развлечения

Firefox справляется с многими форматами мультимедиа, но он способен на большее. Например, есть расширение (**https://addons.mozilla.org/Firefox/2478**) для тех, кто следит за сериями американского мультика Homestar Runner (**www.homestarrunner.com**), есть просмотрщик фотографий Flickr на боковой панели (**https://addons.mozilla.org/Firefox/1135**), а можно интерактивно загрузить, отредактировать и разместить фотографии на главных сайтах-фотоколлекциях инструментом вроде помещенного на сайте **www.picture2life.com** (правда, он пока ждет обновления до версии *Firefox 2.0*).

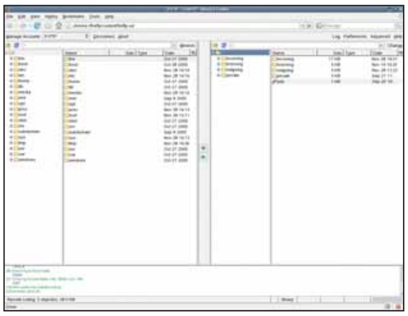
> «Performancing дает вам специальное ПО для публикаций через окно вашего браузера.»

9 С точки зрения чистого удобства, мы просто обязаны были включить в наш обзор Fast Video Download (**https://addons.mozilla.org/Firefox/3590**). Это расширение добавит вам кнопку закачек видеофайлов с наиболее популярных сервисов, среди которых Google Video, MySpace и YouTube. Оно здорово экономит время, и при условии, что вы загружаете только клипы, доступные свободно, смело им пользуйтесь.

10 А как насчет вашего вклада? Если вы ведете онлайн-дневник, вы можете создавать новые публикации в любом браузере; но специально для стандартных CMS-форм ПО не существовало. Расширение Performancing (**www.performancing.com**) позаботилось об этом: оно поддерживает большинство систем, включая Wordpress.com, TypePad, LiveJournal и MSN Spaces, с готовыми конфигурациями для ваших собственных: *Drupal*, *Movable Type*, *Textpattern* и *Roller*. Если ни одна из них не сработает, можно попробовать соединиться через MetaWeblog и Blogger APIs.

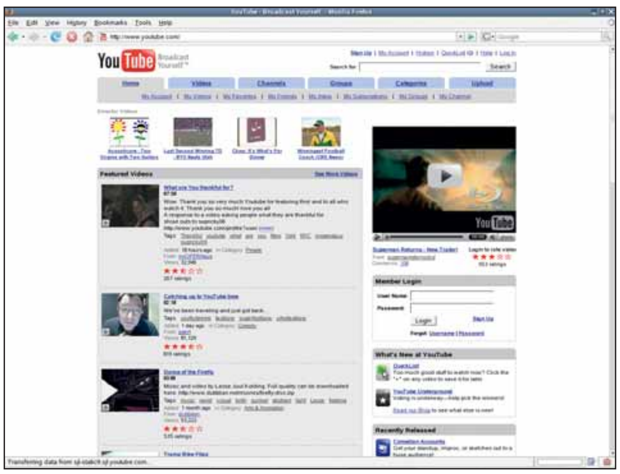
Редактор находится в нижней части экрана и не мешает вам продолжать серфинг, одновременно создавая публикацию. Это полезно при отправке новостей, а также при необходимости куда-нибудь сгонять за справкой. Поддерживаются закладки Delicious и статистика Metrics, сохраняется история публикаций.

Вот и вся наша первая десятка. Жаждете еще? Воспользуйтесь Data Analytics для обработки и извлечения данных из таблиц (**https://addons.mozilla.org/Firefox/2010**), или Deja Click (**https://addons.mozilla.org/Firefox/3262**) для автоматизации работы с браузером, или Fire Encrypter (**https://addons.mozilla.org/Firefox/3208**) для защиты ваших данных, или… или… или… LXF

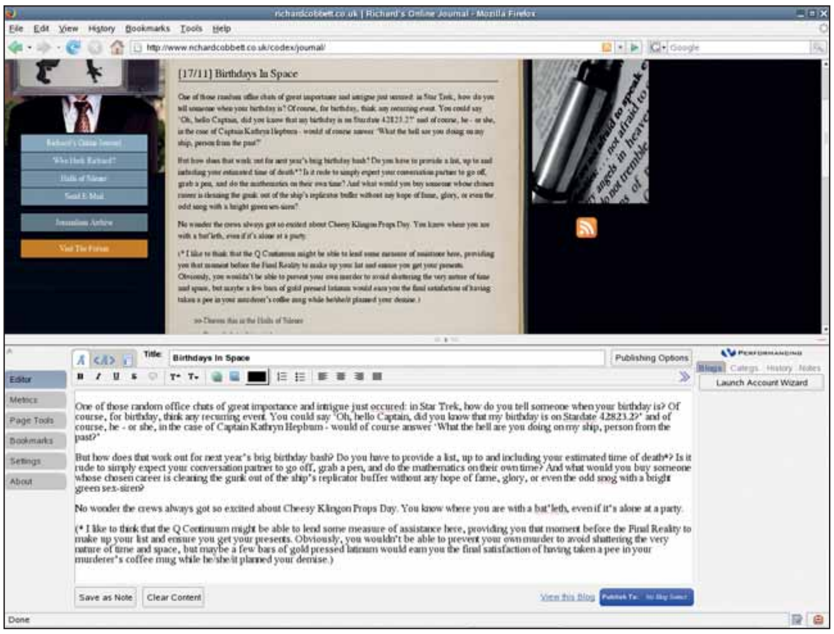


› FireFTP (стр. 35) поддерживает FTP не хуже любого базового браузера.

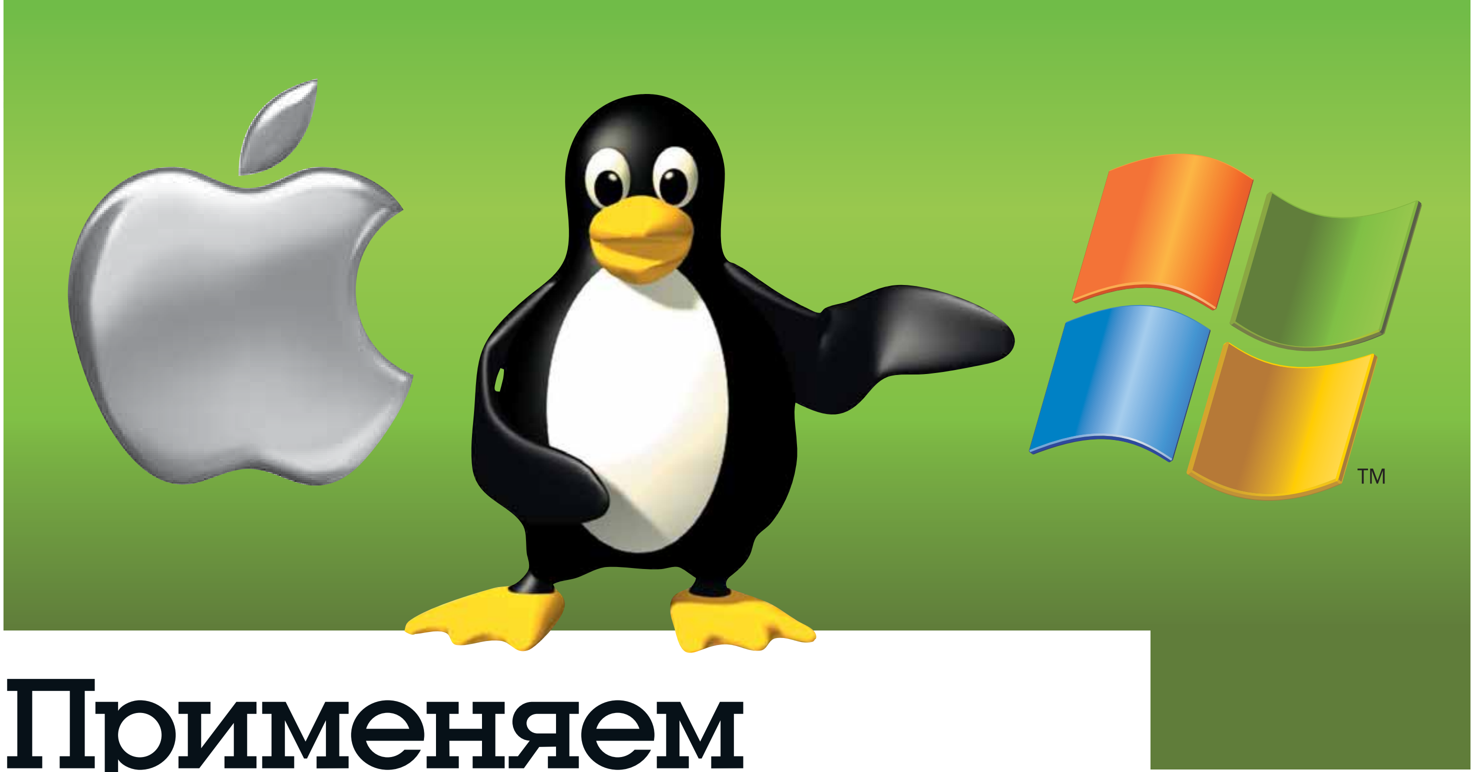




› Fast Video Download немедленно загрузит любое видео. Мы надеемся, вы не будете этим злоупотреблять.



› Отличный редактор блогов Performancing ухитряется держаться в окне *Firefox*, не лишая вас свободы бродяжничать.

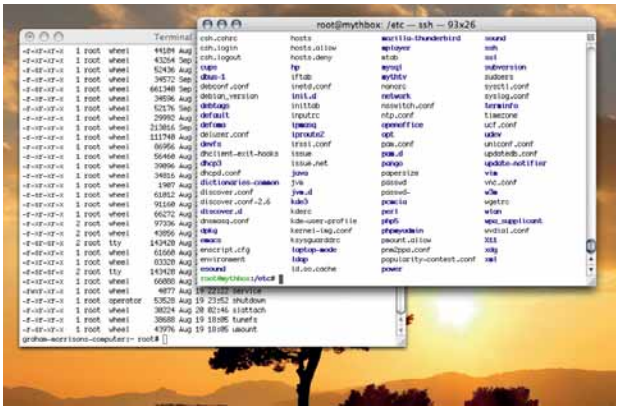


Применяем навыки Linux

Иногда приходится поработать и в других ОС. **Грэм Моррисон** изучает, окажется ли ваш Linux-стаж полезным при работе в MacOS X и Microsoft Windows.

Мы все вложили немало сил и времени, чтобы найти общий язык с Linux. Устанавливая свой первый дистрибутив, настраивая web-сервер в офисе или создавая среду разработки, мы постоянно открывали для себя что-то новое и чему-то учились. И в качестве читателей лучшего в мире журнала о Linux, мы счастливы тем, что наш опыт может постоянно обогащаться.

Но насколько полезен этот опыт? Помогут ли приобретенные знания при работе в других операционных системах? Мы не предлагаем вам тут же ринуться в Windows или OS X, просто хотим узнать, в чем преимущества знатоков Linux на альтернативных ОС.



› В OS X для линуксоидов приятнее всего близость к подсказке от Bash.

Карьерный рост

Спрос на системных администраторов и разработчиков Linux сейчас неуклонно растет, но вакансий для администраторов Windows все-таки больше. И если у вас есть опыт обслуживания *Apache*, он пригодится вам под любой ОС – можете смело упоминать о нем в резюме. То же касается и прикладных программ. Привыкнув редактировать изображения в *Gimp* и работать с электронной почтой в *Evolution*, легко позабыть, что на свете есть целый мир альтернативных приложений, где этот опыт обеспечит вам прочные позиции. Если вы уверенно работаете в Linux и за годы пользования сумели проникнуть в глубины ОС вашего выбора, приятно осознавать, что полученные знания помогут вам при неожиданной смене карьеры.

Даже если какие-то навыки нельзя перенести в другую ОС, все равно знание Linux не было напрасным. Пользователю Linux невозможно удержаться от того, чтобы узнать, как на самом деле работает компьютер. Чем бы вы ни занялись – настройкой соединения по локальной сети, блокировкой порта в брандмауэре или восстановлением жесткого диска – вы всегда узнаете чуть больше о том, как все устроено.

Заглянем вовнутрь

Но главная ценность этого опыта в достижении понимания общих принципов и закономерностей. Например, если почтовый клиент не принимает входящую почту по протоколу IMAP, вы сразу сообразите, что порт может быть просто закрыт на брандмауэре, если когда-то настраивали брандмауэр в Linux – даже если в другой ОС не смогли бы этот порт открыть.

Чтобы разобраться, тяжкой ли покажется работа в других системах, мы сравнили функциональность по умолчанию, предлагаемую Linux и двумя другими ОС. Начнем с Apple OS X, как с более похожей на Linux.

Apple OS X

Пусть эта ОС и графическая, но командная строка здесь тоже полезна.

Apple OS X происходит от разработанной Стивом Джобсом [Steve Jobs] операционной системы Nextstep, которая включала код ядра Mach и Unix BSD. И хотя графическая среда OS X поневоле притягивает к себе все наше внимание, сейчас мы убедимся в том, что в этой ОС много общего с Linux.

Интерфейс пользователя OS X – просто оболочка, под названием Aqua. Как и в Gnome или KDE, полный контроль можно получить парой щелчков мыши благодаря родному интерфейсу командной строки – под псевдонимом Terminal. Вход в него – через папку «Приложения» (Applications), затем «Утилиты» (Utilities), а его значок виртуально идентичен Gnome и KDE. Да и по сути разница невелика.

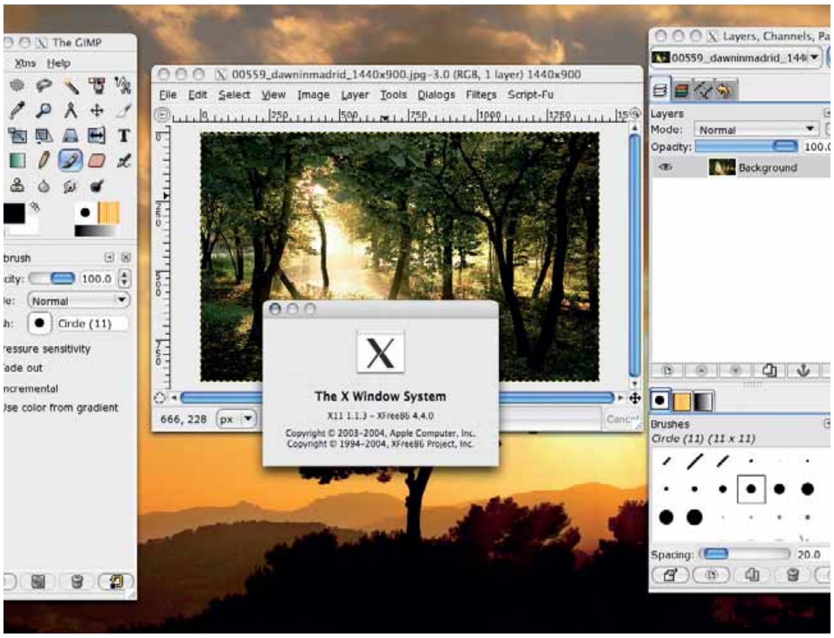
Окно *Terminal* выглядит таким знакомым, потому что командная оболочка OS X – не что иное, как Bourne Again Shell, который мы ласково зовем *Bash*. Поэтому любой, кто пользовался командной строкой Linux, будет в OS X, как дома. Сощурившись на экран, можно даже убедить себя, что никуда и не переходил; это одно из основных преимуществ OS X над Microsoft Windows. Вы сможете делать практически все, чему выучились в Linux, включая создание скриптов и употребление тех же команд.

Основной различие между OS X и Linux кроется в структуре каталогов файловой системы. Правда, большинство файлов конфигурации все еще находится в папке **/etc**, но домашние каталоги пользователей и системные файлы хранятся в совершенно других местах: в папке **/Users** и в папке **/Applications**, соответственно.

Фактор X11

Благодаря близкому сходству Apple и Unix-подобных систем, есть и другие преимущества. Во-первых, разработчикам доступен компилятор GNU (*GCC*). Во-вторых, в OS X присутствует сервер X11; он основан на *XFree86 4.4*, последним выпуском перед печально известным разветвлением, которое способствовало появлению линейки *X.org*, применяемой многими линуксоидами.

Если вы разработчик, вам будет гораздо проще с X11, чем с собственными инструментами Apple. Компания выпускает большое количество кода по собственной открытой лицензии (Apple Public Source Licence) и к большей части оборудования Apple прилагается второй диск с исходными текстами и бесплатной средой разработки, называемой *Xcode*. Основной язык программирования в ней – Objective-C, также доставшийся в наследство от NextStep, но *Xcode* можно настроить и на C++ и даже интегрировать ее со свободно распространяемой версией библиотеки *Qt*. Впрочем, для создания кросс-платформенных решений лучшим вариантом будет открытая среда разработки *Eclipse*, а для быстрой разработки приложений берите что-нибудь вроде *Ruby on Rails*.



› Благодаря наличию X-сервера, Linux-приложения – например, *Gimp* – легко переносятся в OS X.

В качестве отладчика в следующую версию OS X (10.5, кодовое название Leopard) будет включен открытый продукт *DTrace* фирмы Sun. Он станет реальным подспорьем разработчика; вдобавок Apple отправила его на Unix-сертификацию в The Open Group, а это обещает хорошую совместимость и упрощение кросс-платформенных разработок.

Сервер X11 поддерживает аппаратное 2D- и 3D-ускорение. Преимущество этого сервера над «родным» для OS X Aqua – существенное упрощение переноса в OS X приложений из Linux/Unix: это лучший способ заполучить *Gimp*, *OpenOffice.org*, *Ardour* и даже использовать звуковой сервер Jack. Последним пользоваться на удивление просто: установите единственную панель управления, нажмите Play – и готово. При некоторой настройке оборудования пользоваться *Ardour*, Jack и эффектами реального времени даже проще, чем в Linux. Все эти приложения доступны для загрузки в X11-совместимом варианте для оборудования Apple, и будут работать точно так же, как и оригинальные приложения в Linux. »



› **Много новых функций Linux имеют сородичей в OS X: поисковая система *Spotlight* – аналог Beagle.**

Полезные советы: OS X

» **Virtue Desktops** Поддержка виртуальных рабочих столов запланирована на ближайший релиз OS X, но Virtue Desktops позволит вам использовать их уже сейчас. Это приложение с открытым исходным кодом, которое даже позволит скопировать и вставить ваш рабочий стол на грань пресловутого 3D-кубика.
http://virtuedesktops.info

» **Fink** Если вам жизнь не мила без приложений Gnome или KDE, поможет *Fink*. Кроме 2000 поддерживаемых приложений, еще многие тысячи доступны из нестабильных репозиториев.
http://fink.sourceforge.net

» Для установки X-сервера в OS X вставьте первый инсталляционный диск, перезагрузите компьютер и выберите X11 в опциях программы установки. Сервер устанавливается довольно долго (лучше делать это одновременно с установкой OS X), но зато вам будут доступны все сливки Linux-приложений. X-сервер будет запускаться автоматически по запросу приложения. При этом на панели запуска Apple появится логотип X.

Еще один приятный сюрприз – установленный на Mac по умолчанию сервер *Apache 1.3*. Версия web-сервера, используемая в OS X, ничем не отличается от тех, которые поставляются с большинством дистрибутивов Linux, хотя многие уже переходят на версию 2.0. Можно установить и сервер *Samba*, а клиент *Samba* вообще встроен в систему. OS X, может, и не лучшая платформа для серверов, но лучший выбор, если вы хотите обойтись вашим Linux-опытом.

> «X-сервер позволит вам работать с приложениями Linux в OS X.»

Прикладные программы

Apple знаменит дружелюбием к дизайнерам, и если у вас стоит Mac, то не исключено, что вы выбрали его именно из-за *Adobe Illustrator*, *Photoshop* или *InDesign*. Эти пакеты – безусловные лидеры рынка, так что очернять их перед открытыми аналогами было бы не совсем справедливо. Однако, и проприетарные, и открытые приложения строятся по одинаковым принципам. Если вы работали в *Gimp* или *Inkscape*, то без труда освоите *Adobe Photoshop* или *Illustrator*. В основе и тех, и других лежат концепции слоев, фильтров и контуров, и палитры на панели инструментов выглядят почти одинаково. Конечно, некоторое время уйдет на освоение меню, но основные практические приемы работы вы уже изучили в Linux. То же относится и к утилитам рабочего стола. Например, поисковая система *Spotlight* имеет много общего с *Beagle*, а виджеты рабочего стола похожи на *SuperKaramba* от KDE, так что в этом смысле OS X не слишком отличается от хорошо настроенного ПК с Linux.

Microsoft Windows

Linux опережает MS во многих областях – и это ваш козырь.

При попытке применения в Windows опыта Linux сразу же берет тоска от отсутствия интерфейса командной строки: ни *Bash*, ни *Csh*, ни *Terminal* не притаились среди каких-нибудь Utilities. Вместо них мы получаем современный вариант краеугольного камня, на котором строилась империя Microsoft: MS-DOS.

Возможности командной строки Windows (ярлык для ее запуска находится в меню «Стандартные») всегда уступали Unix-эквивалентам, и пользователи Windows годами, вплоть до появления Windows XP, были вынуждены вручную редактировать файлы конфигурации даже для того, чтоб добавить какое-нибудь пошлое запоминание истории команд. Недостает не только интерфейса – знакомых вам команд тоже нет: *ls*, *less*, *ssh*, и уж, конечно, нет *emacs* или *vi*; нет и команды для организации конвейера [здесь автор явно перегнул палку, – прим. ред.]

Подход Windows отражает желание Microsoft упрятать внутренности своей ОС в Панель управления и системный реестр. Правда, можно установить кое-какие команды Unix, но от этого радости мало.

› Привязка приложений типа *Internet Explorer* к определенным каталогам и областям памяти – концепция, не чуждая Linux.



Сервировка

Пусть в Windows вам не пригодятся ни навыки администрирования системы, ни приемы работы в командной строке – все равно вы обладаете знаниями, которые найдут применение. Вообще говоря, администрирование Windows сильно зависит от понимания взаимодействия компьютера с локальной сетью – позади брандмауэра или перед ним – и способа соединения пользователей с Интернетом. Но, как и в OS X, можно установить на компьютере открытый web-сервер и оказаться на родной почве. Можно спокойно запускать из-под Windows и *Apache*, и PHP, и *MySQL*; они весьма похожи на Linux-версии, разница только в файловой системе. А установить их будет даже проще: все сайты разработчиков предоставляют двоичные Windows-пакеты.

Полезные советы: Windows

» **Cygwin** Популярная у разработчиков рабочая среда; под Windows, ближе нее к среде Unix/Linux ничего нет. Устанавливает *X*-сервер, инструменты *GCC* и библиотеки разработчика, а главное, *Bash*! Она не так хорошо интегрирована, как *Fink* на Mac'ах, но все равно отличная штука.
www.cygwin.com

» **Putty** В Windows остро недостает клиента SSH. Имеются коммерческие версии, но *Putty* – очень удобная программа с открытым исходным кодом, работающая и в командном интерпретаторе Windows, и в собственном окне.
www.putty.nl

Установить их просто, но потом никуда не деться от настройки; тут-то ваш линуксоидный опыт и предстанет во всем блеске (правда, надо помнить, что под Windows приложения будут устанавливаться в других каталогах – например, *Apache* помещается в **Program Files**). А вот при работе с собственными продуктами Microsoft уже понадобится знание теории: например, *Microsoft SQL Server* не слишком похож на *MySQL*, хотя язык у них и общий (*SQL*).

Даже продукты типа *Microsoft Server 2003* покажутся незнакомыми среднему линуксоиду: Server 2003 – версия операционной системы Microsoft, поставляемая с мощным серверным оснащением, включая *Exchange* для почты и мгновенных сообщений и *SQL Server* для баз данных. Все как в Windows, только игр нет.

Разработчикам

Многие приложения Linux перехватили инициативу от Windows-аналогов. Например, если вы накоротке с *Evolution*, то легко справитесь с почтовым клиентом *Microsoft Outlook*. То же относится к программе для обмена сообщениями *MSN Messenger*, которая имеет много общего с *Gaim* и *Kopete*. Под Windows, конечно, всегда можно воспользоваться *Firefox*: *Internet Explorer* не покажется вам интуитивно понятным. Многие программы с открытым исходным кодом, в частности, *Gimp*, *Scribus* и *Inkscape*, портированы на Windows, существуют даже открытые Windows-приложения.

Любителям программировать в Windows будет житься неплохо: здесь имеются прекрасные среды разработки, например, бесплатно распространяемые версии одной из лучших IDE – *Visual Studio*. И с Visual Basic, и с C++ вам будет здесь хорошо. Основной язык и платформа Microsoft Windows – C#/.NET – портированы в Linux в рамках проекта Mono (см. стр. 22), но *Visual Studio* – настолько удачное средство для подобных разработок, что можно всерьез задуматься об использовании его вместо Linux. Если же вам нужна кросс-платформенная совместимость, то, как и в OS X, можно использовать одну из открытых сред разработки, например, *Eclipse*.

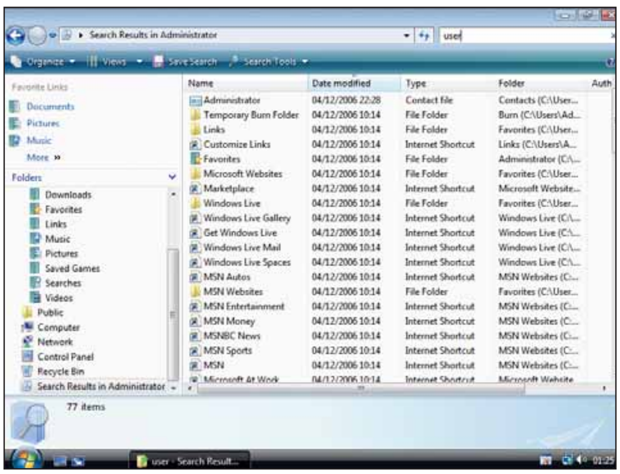
Перспективы Vista

Microsoft проработала пять лет, решая накопившиеся с Windows проблемы, но догнать Linux не удалось – разве что вы не слишком обеспокоены сетевой безопасностью. По-прежнему можно работать в командной строке, появилась *Windows Desktop Search* – некое подобие локальной поисковой системы типа *Spotlight* от Apple или *Beagle* для Linux.

Главные изменения заключаются в том, что Microsoft наконец ввела ограничение прав доступа для некоторых приложений. *Internet Explorer* запускается в «песочнице» по типу 'root jail', когда процесс имеет право доступа только к определенной директории или к заданной области памяти. Существенные изменения произошли и в управлении правами пользователя. Теперь для установки и удаления программного обеспечения или изменения конфигурации системы пользователь должен будет ввести пароль администратора. Именно так и работает Linux.



› Настройка Vista может потребовать пароля администратора.



› Осваивая любую новую ОС, первым делом знакомятся с ее файловой системой.

Вперед, к победе!

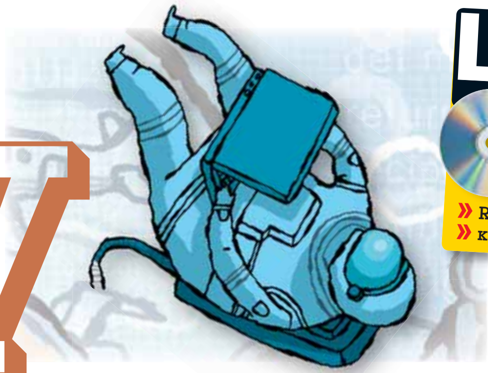
Если вам нужно переходить на другую ОС, или вы хотите слегка отдохнуть от Linux, лучшим вариантом, несомненно, будет OS X. Небольшие добавки: клиент SSH, приличный терминал, а также X11 – помогают освоиться здесь гораздо быстрее, чем в Windows. OS X дает наилучшую возможность применить ваши нажитые тяжким трудом Linux-таланты.

Нельзя сказать, что под Windows эти таланты окажутся не у дел. Windows просто создает лишний уровень абстракции по сравнению с подходом Linux. Проблемы все те же, но их решение в Windows отстоит дальше от первопричин. Самое ценное, что вы получаете от Linux – это глубинное понимание происходящих процессов. Пусть процесс от вас скрыт, но причины проблем остаются теми же в любой ОС. Если вы сталкивались с ними в Linux, вы распознаете их везде. Выходит, незачем и покидать Linux. LXF

«Какие технологии, к которым я привык в Linux, можно использовать в OS X или Windows?»

Технология	OS X	Windows
X11	✓	☒
Bash	✓	☒
GCC	✓	✓
Apache	✓	✓
PHP	✓	✓
MySQL	✓	✓
SSH	✓	☒
.NET/Mono	☒	✓

Ruby без Rails



LINUX FORMAT на DVD
» Ruby 1.8.5
» код из статьи

Вы слыхали о Ruby on Rails – а сейчас Майк Сондерс расскажет вам о его основе, Ruby, ярком примере современного языка высокого уровня…

Вот и еще одна статья из цикла об «экзотических» языках программирования. Язык Tcl, с которым мы познакомились месяц назад, очень подходит для разработки оконных приложений с графическим интерфейсом, а сейчас мы займемся Ruby, предназначенным для создания серверных и сетевых приложений. Заскучали? Напрасно. Сейчас объясним, почему.

Мы постараемся не погрязнуть в технозе: на то есть подробнейшая документация. Вместо этого мы начнем с основ языка, затем напишем простенькое приложение, а в завершение – полноценный web-проект. Вы уже знакомы с азами программирования? Отлично! Читайте дальше, и вы поймете, что Ruby достоин вашего внимания.

Многие из нас слышали о Ruby в контексте Ruby on Rails, современного программного каркаса [framework]. На его основе сейчас разрабатываются крупные сайты (например, Basecamp, Jobster или 43 Things), и он поставляется с новой версией OS X, Leopard. На ринг Ruby вышел совсем недавно, и тем удивительнее тот факт, что его разработка ведется с 1993 года!

> «Легкость операций над текстом помогла Ruby завоевать расположение разработчиков сайтов.»



www.ellywalton-illustrations.com

Детище Юкихиро Мацумото [Yukihiro Matsumoto] увидело свет в 1995 году. Ключевой особенностью языка стал принцип освобождения программиста от черной работы. Зачем продираться через заковыристый синтаксис, если требуется сделать что-то простое?

Проработав более двух лет на C++ и найдя, что он все еще сталкивается с неприятными сюрпризами, Мацумото решил создать язык, максимально оправдывающий ожидания – чтобы в нем программы делали именно то, что интуитивно предполагалось, и отсутствовали тупиковые случаи. «Прежде всего надо думать о людях», провозгласил Юкихиро, «о том, как они мыслят о программе и как представляют себе действия машин. Люди – хозяева. А машины – работники».

С историей – все, теперь о кодировании: Ruby называют «мультипарадигменным» [multiparadigm] языком. Это жутковатое слово означает, что язык включает и объектно ориентированный подход, и процедуры с функциями. Ruby – интерпретирующий язык, и его синтаксис близок к Python и Perl, поэтому он осваивается без труда.

Освоимся

Для начала нам понадобятся интерпретатор языка Ruby и интерактивный интерпретатор *irb*, поставляемые с большинством дистрибутивов. Последнюю версию этих программ вы найдете в разделе **Разработка** нашего DVD.

Вызовем интерпретатор, набрав **irb** в командной строке. На экране появится строка приглашения – можно с ходу вводить операторы языка. По традиции полагалось бы вывести “Hello world” – «Здравствуй, мир», но это старо как мир... Будем оригинальны:

```
puts “Goodbye moon!”
```

Вроде все ясно: команда puts просто выводит на экран текстовую строку. Но в Ruby абсолютно все, даже строковые константы, – это объекты. Может, оно и странно, зато позволяет делать штуки вроде

```
“BlaBlaBlaa”.length
```

С виду чушь, а на выходе печатается **10** – вполне настоящая длина строки. Или, например,

```
“Mmm, donuts”.index(“d”)
```

возвращает **5**, позицию символа d в заданной строке. (Учтите, нумерация в Ruby начинается с нуля, а не с единицы.)

Итак, Ruby – страна объектов, где программировать значительно проще. Вдобавок, это еще и язык с динамической типизацией: переменные не нужно объявлять до их использования. Вот пример:

```
puts “Do you love Ruby yet?”
answer = gets
puts “You answered ” + answer
```

На экран выводится вопрос, потом объявляется переменная answer, и в нее тут же записывается результат команды gets (ввод строки с клавиатуры). Наш ответ затем выводится на экран.

Заодно продемонстрирована операция над переменными: строки можно складывать (+) и даже умножать:

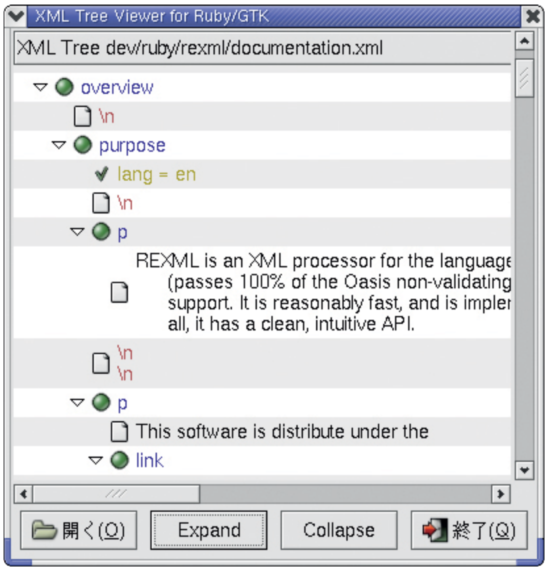
```
mystring = “Some” + “text”
puts mystring
anotherstring = “Wowzers” * 10
puts anotherstring
```

Сначала две строки объединяются, и результат выводится на экран. Затем строка anotherstring заполняется десятью копиями строки Wowzers, и результат также выводится на экран. Именно эффективные функции обработки текста обеспечили Ruby место под солнцем в мире разработчиков сайтов. Неплохо знать и метод chomp (вы могли встретить его в Perl): он убирает из строки последний символ, перевод строки (\n), который автоматически добавляется командой gets при вводе. Метод понадобится в тех случаях, когда нужно преобразовать введенную строку в число, т.е. в «голые цифры»:

```
x = gets.chomp.to_i
```

Мы создали переменную x, ввели строку командой gets, откинули у нее символ перевода строки с помощью chomp и методом to_i преобразовали ее в числовое значение, которое затем присвоили x. Тип переменной x можно проверить:

```
x = gets.chomp.to_i
```



› *Ruby-Gnome2* **поможет создать приложение для Gnome.**

```
x.is_a?(Integer)
x.is_a?(String)
```

Это показывает, что x – число, а не строка. Но мы можем немедля использовать x и как строку, если надо – благодаря динамической типизации. [Вопросительный знак является допустимым символом и часто используется в Ruby в именах методов, которые «отвечают на вопрос», – прим. ред.]

Условия, циклы и функции

Разобравшись с переменными и вводом/выводом строк, перейдем к управляющим конструкциям языка. Каждый из приведенных ниже примеров нужно предварительно сохранить в текстовом файле и запускать командой **ruby <имя_файла>**. Сохраните следующий пример в файле **test.rb** и запустите командой **ruby test.rb** – он иллюстрирует использование условного оператора if и кодовых блоков:

```
puts “How many kittens do you have?”
x = gets.chomp.to_i
if x > 0 # More than zero kittens
    puts “You have ” + x.to_s + “ kittens”
else
    puts “Aww, you have no kittens!”
end
```



Мы запрашиваем у пользователя строку, затем уже известным нам методом to_i преобразуем ее в число. Далее оператор условия if сравнивает введенное число с нулем. Если условие в операторе if истинно, число преобразуется в строку с помощью метода to_s, чтоб вписать его в сообщение, и сообщение выводится его на экран.

Если же пользователь ввел **0**, выполняются команды, расположенные после else, то есть выводится другое сообщение. Блок кода завершается командой end. Между прочим, для лучшей читаемости текста мы применили отступы – можно их вводить клавишей табуляции, а можно и просто пробелами. Все как в других языках!

Справа от оператора if помещен комментарий (в языке Ruby они предваряются символом #).

С циклами тоже проблем нет:

```
10.times do
    puts “Ruby rocks”
end
```

Все ясно – десять раз выводится строка! Оператор while ничуть не сложнее:

```
x = 0
while x != 15
    puts “Please enter 15”
```

»



```
  x = gets.chomp.to_i
end
```

Этот код требует у пользователя ввода строки до тех пор, пока не получит число 15. Тогда он покидает блок while/end и продолжает выполнение кода (правда, в данном случае выполнять уже нечего).

Программирование на языке высокого уровня обязательно подразумевает модульность, так что займемся процедурами. Как мы уже говорили, Ruby – объектно-ориентированный (ОО) язык, аналогично C++, C# или Python. Но даже если вы знакомы только с обычным C, можете не бояться ОО-сложностей, по крайней мере, на начальном этапе. Рассмотрим использование функций:

```
def doubler(value)
   value = value * 2
   return value
end
puts "Enter a number"
x = gets.chomp.to_i
y = doubler(x)
puts y.to_s
```

Первая половина скрипта – описание функции, она не будет исполняться при запуске. Функция doubler умножает число value (аргумент функции) на два и возвращает результат. Из кода видно, как используется функция. (Обратите внимание, что функции должны быть объявлены до их использования, иначе Ruby выдаст ошибку.)

Выполнение программы начинается с команды puts: у пользователя запрашивают число, и оно передается в качестве параметра функции doubler. Результат, возвращаемый функцией, записывается в переменную y. Наконец, значение переменной y выводится на экран. Создание объектов в Ruby выходит за рамки данной статьи; хотите узнать больше – обратитесь на сайты, указанные во врезке «Ссылки» на этой странице.

Работать чисто

Обработка исключительных ситуаций в Ruby блаженно проста, например:

```
puts "Divide 5 by what?"
x = gets.chomp.to_i
begin
   y = 10 / x
rescue ZeroDivisionError
   puts "Eeep, divide by zero!"
end
puts y.to_s
```

Мы берем у пользователя число, потом входим в блок begin/end с разделом rescue внутри. Если пользователь, допустим, введет **5**, в переменную y записывается частное от деления **10** на **5** и результат выводится на экран.

А вдруг он введет ноль? Возникнет ошибка. В таких случаях происходит возврат в командную строку и вывод сообщения, над которым вы и будете ломать голову. Но мы не зря ввели обработчик исключения. Возникшая исключительная ситуация (деление на ноль) будет зафиксирована, и выполнится оператор puts внутри обработчика исключения, который выведет на экран сообщение об ошибке.

> «Со знанием Ruby создать web-сервер можно быстрее, чем произнести "/etc/apache/httpd.conf"...»

Ссылки

- » www.ruby-lang.org Домашняя страница Ruby.
- » http://rubyforge.org Библиотеки и программы.
- » www.rubycentral.com/book Отличный учебный курс по Ruby.
- » http://tryruby.hobix.com Онлайн-курс.

Обработка исключений исключительно удобна: она спасает вас от ползания по коду в поисках проблемы и вставки проверок куда ни попадя. Ваш опрятный и компактный код отловит множество видов ошибок при исполнении, открытии файла и т.д.

Создаем web-сервер

Применим полученные знания и напишем собственный… web-сервер! Экстрим? Да нет, в Ruby это совсем несложно, благодаря библиотеке socket. Конечно, для *Apache* наш сервер конкурентом пока не станет, но зато продемонстрирует возможности Ruby. И вообще, зачем устанавливать и настраивать мощный сервер, если вам всего-то нужно хранить пару web-страничек для доступа к ним по локальной сети? Вы напишете собственный сервер на Ruby быстрее, чем произнесете "/etc/apache/httpd.conf"!

Сохраните следующий код в файле **webserver.rb** в вашем домашнем каталоге. (Если вам лень набирать, возьмите его из раздела **Magazine/Ruby** нашего DVD.) Затем создайте в этом же каталоге две HTML-странички: **index.html** и **test.html**, с произвольным содержимым, лишь бы выводилось в браузере.

```
require 'socket'
webserver = TCPServer.new('127.0.0.1', 7125)
while (session = webserver.accept)
  session.print "HTTP/1.1 200/OK\r\nContent-type:text/html\r\n\r\n"
  request = session.gets
  trimmedrequest = request.gsub(/GET\ \//, '').gsub(/\ HTTP.*/, '')
  filename = trimmedrequest.chomp
  if filename == ""
    filename = "index.html"
  end
  begin
    displayfile = File.open(filename, 'r')
    content = displayfile.read()
    session.print content
  rescue Errno::ENOENT
    session.print "File not found"
  end
  session.close
end
```

Запустите программу командой **ruby webserver.rb** и проверьте, как она работает. Для этого в адресной строке браузера введите **http://127.0.0.1:7125**. Если все нормально, вы должны увидеть содержимое странички **index.html**! Соответственно, набрав **http://127.0.0.1:7125/test.html**, вы увидите содержимое странички **test.html**. Код говорит сам за себя, но мы все равно разберем его подробно.

Начинается код с команды require, подключающей внешние библиотеки. В данном случае мы используем библиотеку *socket* – это набор процедур, сильно упрощающий сетевое программирование (см. врезку «Модные модули» справа). Следующая строка создает объект сервера на локальном IP-адресе 127.0.0.1 и порте 7125. Номер порта может быть любым, но для использования портов до тысячного (например, 80) потребуются права администратора системы.

У нас получился сервер локальной сети; а мы хотим, чтоб с ним могли работать web-браузеры. Цикл while обрабатывает запросы к серверу, пока в окне терминала не нажать клавиши Ctrl+C. Внутри этого цикла сервер принимает запрос клиента и посылает браузеру стандартный ответ из двух строк: сообщение, что запрос принят, и HTML-заголовок. Символы \r и \n означают «возврат каретки» и «перевод строки» соответственно.

Затем обрабатывается запрос, полученный от браузера. Обычно он выглядит примерно так: **GET /filename.html HTTP/1.1**. Нам нужно определить в нем имя запрошенного файла. Сохраним строку запроса

в переменной request и извлечем оттуда имя файла методом gsub: он отделит от строки части **GET** и **HTTP/1.1** при помощи регулярных выражений. Gsub напоминает командную утилиту sed – быстрый способ заместить или отделить части строковой переменной.

От полученного имени файла с помощью метода chomp отделяется последний символ перевода строки, после чего результат сохраняется в переменной filename.

Большинство web-серверов, если имя файла не указано, по умолчанию загружают **index.html**. Блок **if** так и делает: если строка с именем файла пуста, переменной filename присваивается **index.html**. Можно, конечно, использовать и другой файл.

После этого (внутри блока begin/end) мы открываем полученный файл на чтение и записываем в переменную content его содержимое, прочитанное с помощью функции read(). Содержимое файла отсылается браузеру, методом session.print. А вдруг такого файла не найдется? Тогда rescue отправит браузеру сообщение об ошибке.

Мысли напоследок

Вот сервер и готов. Удивительно просто, правда? Конечно, пока он обрабатывает только HTML-файлы – но были бы кости, мясо нарастет. Главное, что это пример применения Ruby в реальной задаче, и по нему видно, что код получается чудесно-простой и легко читаемый.

В Ruby еще много чего можно исследовать, и так как Ruby on Rails продолжает набирать популярность, вы об этом языке, несомненно, еще услышите. Ясный и разумный синтаксис Ruby лучше всех прочих поможет разобраться с объектно-ориентированным программированием тем, кто привык к C или Basic. Внизу слева находится врезка «Ссылки», там вы найдете и учебники, и прочую информацию. А напишете нечто крутое – сообщите нам на форум www.linuxforum.ru. Как знать, может, вашей будущей программы не хватает на нашем DVD... LXF

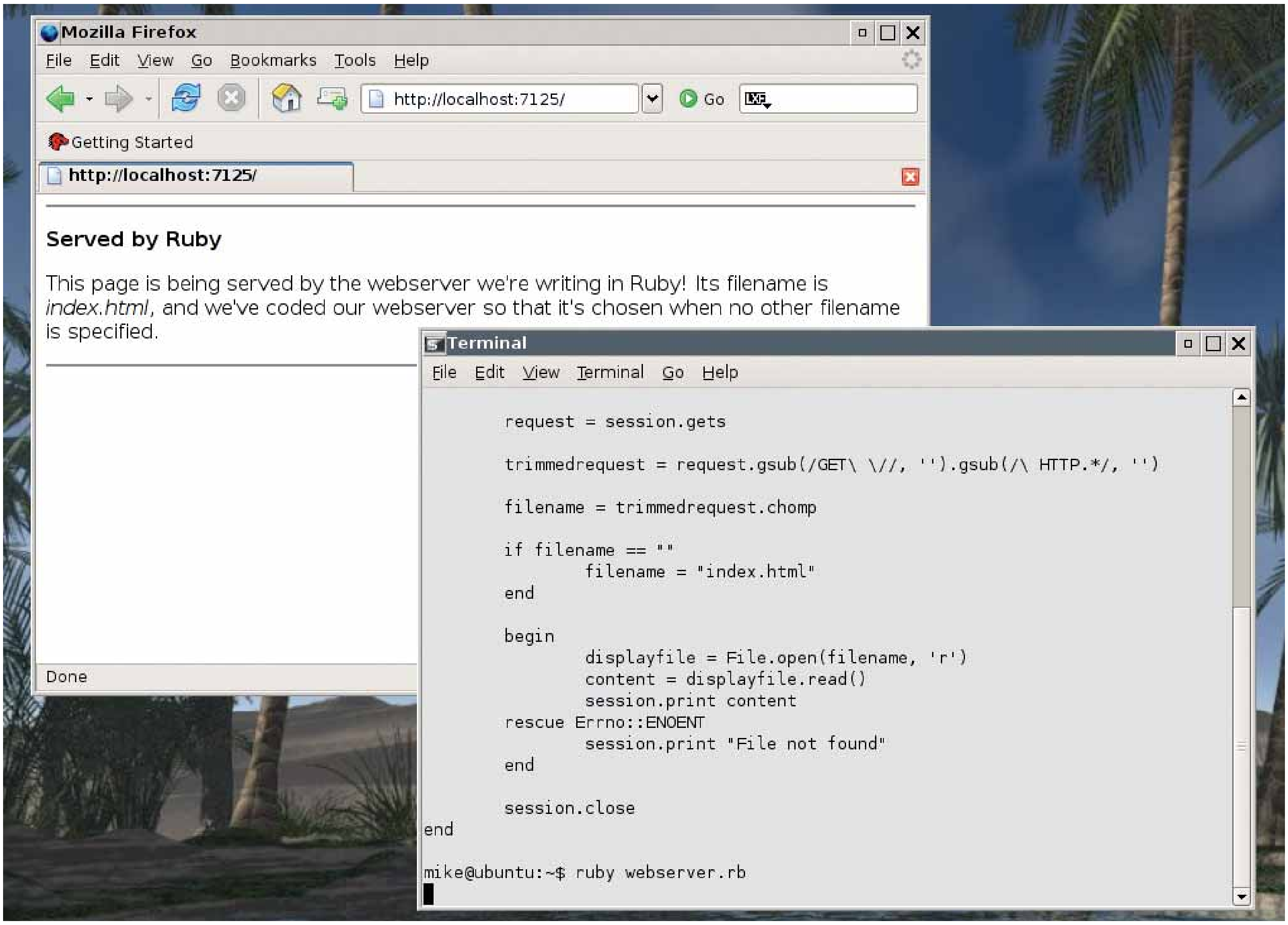
Модные модули

Ruby поставляется с набором модулей, расширяющих возможности языка, подобно библиотекам C модулям Python. В зависимости от используемого дистрибутива и версии Ruby их расположение может различаться, например, в Ubuntu 6.06, они расположены в каталоге **/usr/lib/ruby/1.8/i486-linux/**. В стандартную поставку входят модули для разработки сетевых приложений, сжатия данных с использованием *zlib*, консольных приложений на базе Ncurses и многие другие. Их функциональность добавляется в программу строкой require, как в нашем примере с сервером.

Дополнительные модули можно загрузить через систему RubyGems, аналогичную CPAN для Perl. Разработаны модули для взаимодействия с базой данных MySQL, разработки приложений для Gnome и KDE и т.д. Похоже, недолго ждать времени, когда на Ruby можно будет сделать почти любое приложение рабочего стола. За дополнительной информацией обращайтесь на сайт RubyForge (см. врезку «Ссылки»).



› Сайт RubyForge: кладезь библиотек и примеров приложений.



› Результат нашего проекта – простой, но полнофункциональный web-сервер!

» Через месяц Мы рассмотрим самые странные из существующих ныне языков.

Что за штука... Python 3000?

Переписать заново язык программирования, особенно такой популярный, Python – вещь непростая. **Ник Вейч** размышляет, настало ли время обновления...

Python 3000? Я думал, это сказки.
Хм! Ну да, вроде того. Когда Гвидо ван Россуму [Guido van Rossum, создатель Python] намекали, что не худо бы включить очередную полезную для языка, но сложную в реализации штуку, он обычно отвечал, что непременно включит ее в Python 3000 (или py3k). Так что это что-то вроде Святого Грааля для Python, если хотите.

Но я так понял, что сказка становится былью?
О да, Гвидо впервые заговорил о py3k как о реальном проекте еще в 2000 году. А сейчас это реальная версия, и когда будет закончена – станет выпуском Python 3.0. Данный релиз, как обещалось все эти годы, будет содержать фундаментальные изменения. Изменение первой цифры номера версии позволяет провести радикальные переделки, если надо – вплоть до нарушения обратной совместимости...

Стоп-стоп-стоп... так что, мой код на Python 2.x не запустится под Python 3000, если не переписать? Разве это не ужасно?
И да, и нет. Приятно, конечно, когда старый код продолжает работать в новой среде. Но некоторые планируемые изменения в языке делают обратную совместимость непрактичной. Чтобы что-то заработало лучше, оно должно работать иначе. Гвидо сказал, что Python 3000 – это «единственный шанс сделать Python эффективнее». К тому же самое время починить множество мелких поломок.

Хмм... Вы меня не убедили.
Ладно, вот вам пример из целочисленной математики. Если X равно 1, а Y равно 2, каков будет результат X/Y?

Не ожидал тут испанской инквизиции...
[Демонический хохот] Вот и никто не ожидает... Не увиливайте от ответа. Вы его знаете?

Ну, это вопрос с подвохом. Целочисленная математика, говорите? Мой тяжкий опыт отладки где-то первой дюжины программ на Python говорит, что 1/2 вообще-то равно 0.
Верно! Ответ правильный, но большинство людей такого не ожидает. Большинство ожидает, что 1/2 вернет 0,5. Это еще тяжелее усвоить, когда вместо чисел в выражении используются переменные, в особенности начинающим программистам.

Но если это изменить, то угробится огромный объем существующего кода...
Да, это так, но не говорите, что вас не предупреждали – в буквальном смысле слова! Если вы используете оператор / для деления целого числа на целое, то получите предупреждение. Во всяком случае, если запустите Python с ключом –Qwarnall. Фактически, необходимым условием изменения работы Python является выдача предупреждений о том, что такой код скоро канет в Лету, еще в версии 2.x.

Пусть так, но ведь пострадает множество людей, у которых есть тонны готового кода. Нет ли способа сделать его Python 3000-совместимым?
Скорее всего, автоматический конвертор сделать не получится. Но что мешает продолжать запускать этот код на Python 2.x? От добра добра не ищут...

Выходит, разработка ветки Python 2.x будет продолжаться?
Да, пока это будет нужно людям. Никто не собирается бросать поддержку текущей ветви Python. Гвидо планирует, что разработка Python 2.x не прекратится вплоть до выхода Python 3.2, а то и дольше (при этом он не думает, что паузы между релизами py3k окажутся меньше, чем были между релизами 2.x). Последние версии ветви 2.x скорее всего будут содержать некоторые элементы py3k.

> «Смена первой цифры версии позволяет серьезные переделки, вплоть до отмены обратной совместимости.»

И как это будет работать?
Ну, это зависит от конкретного элемента... Некоторые новые возможности можно реализовать более чем одним способом. Это не слишком похоже на Python, но может упростить переход на Python 3000.

Ладно, вы меня уговорите, если мы наконец получим работающий тернарный оператор.
А он уже есть. Хотя его ожидали только в py3k, он был реализован еще в сентябре как часть Python 2.5. Так что Гвидо не так уж плох.

Ого! Что еще можно стянуть из py3k?
Во-первых, будет сделано множество изменений в подсистеме ввода-вывода, она станет гораздо проще. Например, когда разрабатывался Python, Unicode еще не было, но ведь он решает много проблем и вообще хорошая штука. Так что в Python 3000 весь текст будет выводиться в Unicode.

Кстати о вводе-выводе: похоже, оператора print больше не будет...

Что?! И как, по-вашему, я буду выводить данные из скриптов?
Не волнуйтесь: print будет реализован как функция. Ведь на самом деле он оператор только потому, что записывается как оператор. Сделать его функцией гораздо разумнее, примерно так: print(x,y,z).

Ничего подобного! Придется вводить кучу скобок на каждом шагу!
Ну, в целом это сэкономит ваше время. Представьте отладочный код, натыканный повсюду в вашей программе и выводящий много-много экранов данных. Когда нужда в нем отпадет, и вы захотите преобразовать его в вызовы функции сохранения в журнале, достаточно будет воспользоваться функцией поиска и замены строки или даже подменить в модуле, определив функцию для замены (хотя этого делать не рекомендуется: такой код плохо читается).

Короче, в Python 3000 я не смогу ни вычислять, ни показывать результаты привычным способом. Прекрасно. Чем еще порадуете?
О, осталось совсем немного. Всего-навсего удаление лямбда-функций, reduce() и других конструкций функционального программирования...

Во как! Знать бы еще, что такое лямбда-функция.

Лямбда-функция – это короткая функция, определяемая прямо на месте использования. Вообще-то приговор ей отсрочен. Но она по-прежнему является занозой в чистом и ясном коде на Python. Однако Гвидо утверждает, что reduce («свертка», функция, выполняющая некоторую операцию над списком и возвращающая скаляр) обычно используют либо в тривиальных случаях, либо в непознаваемых. В интересах ясности кода гораздо лучше будет использовать явный цикл.

Погодите. Я знаю, что Гвидо – Пожизненный Добрый Диктатор, но не слишком ли это навязчивая опека – указывать людям, как писать их программы, а?

Python всегда и был устроен именно таким образом. Вы взгляните на правила форматирования! Один из основополагающих принципов – тот, что очень, очень трудно, если не невозможно, написать непонятный код на Python. Для Python 3000 это будет еще более верно!

Видимо, вопрос только вот в чем: когда выпустят Python 3.0?

Текущий план подразумевает выпуск в 2007 году, но окончательную версию вряд ли стоит ожидать до 2008 года.

Ну-ну... Вы хотите сказать, он выйдет раньше, чем Perl 6?

Гвидо твердо решил, что ситуация будет не такая, как с Perl 6. Это одна из причин, почему py3k не будут переписывать с нуля. А другая – Python 2, в конце концов, не так уж плох!

Слыхал я такое и раньше. Вы сами сказали, он уже много лет говорил, что берется за этот проект...

Да, но сейчас он вполне серьезен. Честно. Python сейчас важнее, чем когда бы то ни было – его используют везде. Вспомните об утилитах администрирования в Fedora, или о той куче программ, на основе которых работает Google.... потому-то они и наняли Гвидо. Он говорит: «Назовите сторонний модуль для Python, и, скорее всего, в Google найдется человек, который его использует». А как вы думаете, сколько времени они оставляют Гвидо на разработку Python?

Мм...

Ну попробуйте хотя бы предположить...

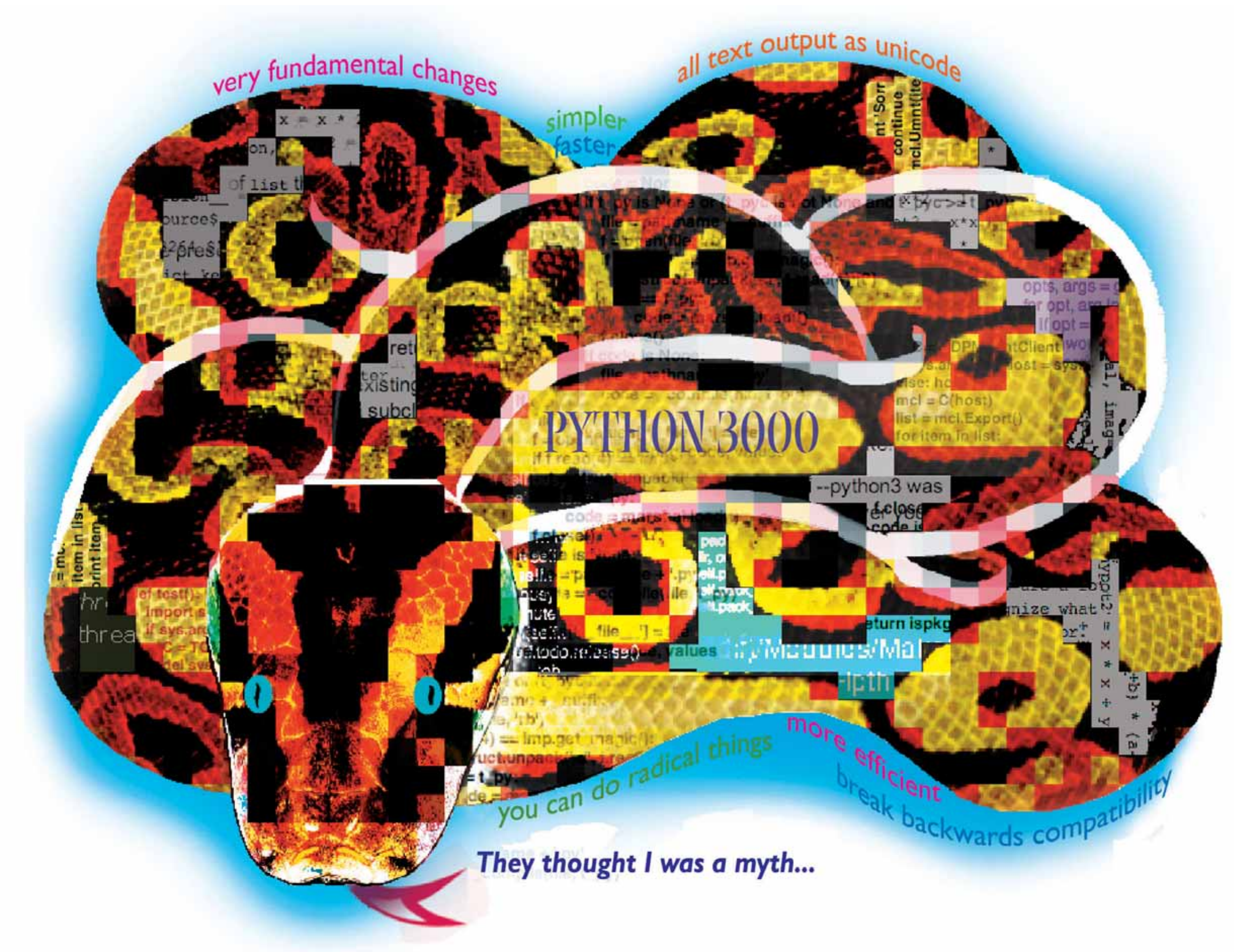
Ноль?

Сейчас как схвачу подушку…

Погодите! Пока вы не начали, скажите, как мне быть в курсе стремительного развития py3k?

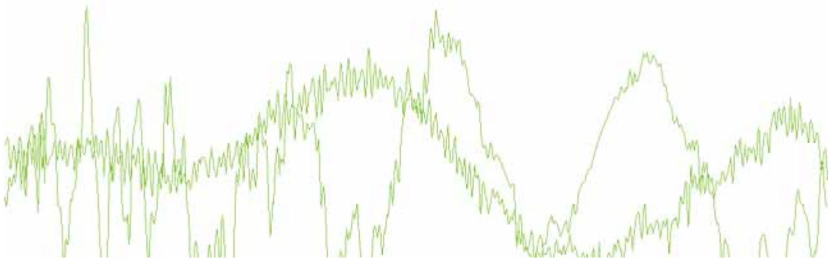
Не язвите. Посмотрите www.python.org/dev/peps/pep-3000 для начинающих и уверуйте! LXF

> «Я знаю, что Гвидо – Пожизненный Добрый Диктатор, но не слишком ли навязчива эта опека?»



Краеугольный

ЧАСТЬ 2: Петр Семилетов нехотя выпускает из рук виртуальные барабанные палочки и берется за клавиатуру, чтобы рассказать вам о драм-машинах.



Важной частью среды обитания музыкальных программ является звуковой сервер Jack (http://jackaudio.org). Некоторые (однако не все) программы могут выводить звук только с его помощью. Что же такое Jack? Грубо говоря – это виртуальный микшер для перенаправления звуковых потоков, исходящих от программ-клиентов. Такие клиенты могут иметь входные и выходные порты. Jack – прослойка между ними. Вы можете запустить какой-нибудь синтезатор или плейер, направить из него звук через Jack на виртуальную стойку с эффектами, а со стойки послать обработанные звуковые данные в еще какую-нибудь программу-клиент. Для управления всем этим используются графические интерфейсы, такие как *QJackCtl* или *QJackConnect*.



› Рис. 1. Главное окно *QJackCtl*.

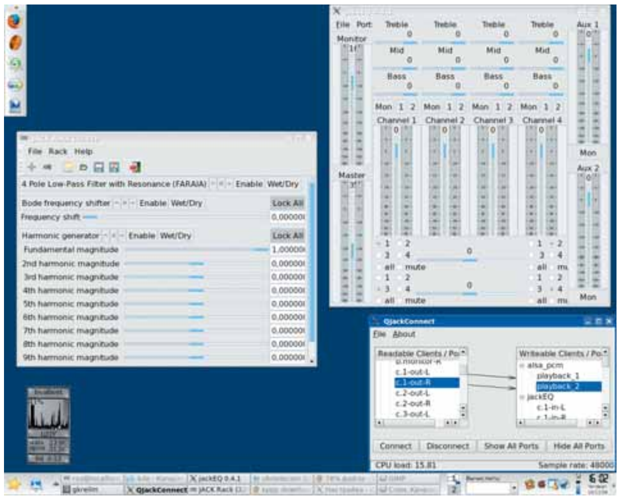
Однако, если вы не собираетесь заниматься таким «диджейством», а хотите просто работать с Jack и выводить на него звук из используемой вами программы (а Jack, в свою очередь, будет передавать его на выходной порт звуковой карты посредством ALSA или другой звуковой подсистемы), то достаточно просто запустить Jack – в простейшем случаев это делается командой:

```
jackd -d alsa
```

Параметр **-d** задает тип звуковой подсистемы. В Linux это, в большинстве случаев, ALSA. Гораздо реже используется OSS – ныне разве что для профессиональных дорогущих звуковых карт. Каждой подсистеме можно передавать дополнительные параметры. Их список для ALSA вы получите, если дадите в консоли команду:

```
jackd -d alsa --help
```

Важным аспектом в использовании Jack является то, что Jack не добавляет временную задержку при перенаправлении сигнала.



› Рис. 2. *JackEQ, JackRack* и *QJackConnect*

На этой иллюстрации вы видите программы: *JackEQ*, *JackRack* и *QJackConnect*. *QJackConnect* служит для того, чтобы связывать программы-клиенты Jack, соединяя их виртуальные входные и выходные порты. Виртуальная стойка с плагинами-эффектами *JackRack* получает звуковой сигнал с порта ALSA PCM capture (им может быть, например, порт Line-In звуковой карты). От *JackRack*, пройдя обработку подключенным к стойке плагинами, звуковой поток направляется на эквалайзер *JackEQ*, а с него – на выходной порт ALSA PCM. Отметим, что на картинке эти связи не совсем очевидны, поскольку стрелки, их представляющие, из-за прокрутки списков с портами исчезают.

Я обратился к теме Jack потому, что программы, о которых речь пойдет далее, так или иначе могут использовать Jack, а программы, о которых я расскажу в третьей части статьи, будут работать только с Jack.

В прошлый раз мы говорили о трекерах, а сегодня речь пойдет о другом классе программ, эдакой промежуточной нише между ними и навороченными цифровыми аудиостанциями (DAW – Digital Audio Workstation). Если такое сравнение допустимо, то трекеры – это холодное оружие, драм-машины и программные синтезаторы/сэмплеры/секвенсеры (не плагины) – это пистолеты-автоматы, а DAW – это уже артиллерия.

Музыку, созданную в некоторых программах «промежуточной» ниши, можно использовать как самостоятельную, либо же микшировать с партиями в других программах. Сегодня я расскажу о двух, по моему мнению – наиболее ярких представителях программ такого вот «среднего» класса.

Hydrogen

Web: www.hydrogen-music.org

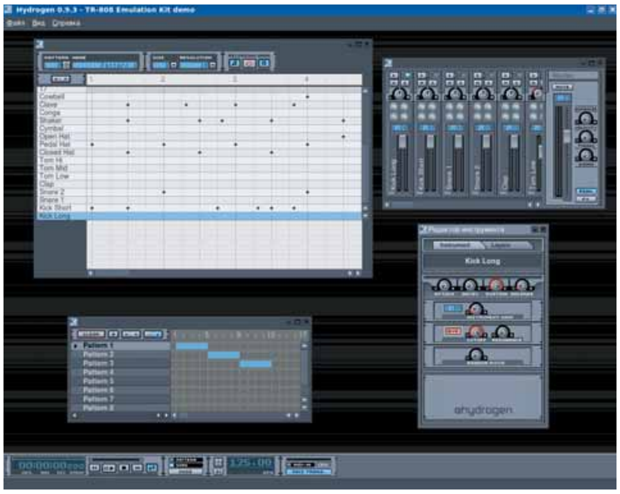
В отличие от большинства своих Windows-аналогов, которые созданы

» Месяц назад Мы совершили экскурс во времена Amiga, рассмотрели Schism и другие программы-трекеры.

камень

в виде плагинов, *Hydrogen* – это драм-машина (барабанная машина), реализованная как отдельная, самостоятельная программа. В качестве библиотеки для графического интерфейса разработчики выбрали *Qt*, а своей темной цветовой гаммой интерфейс *Hydrogen* напоминает *Adobe Premiere Pro 2* и *Nuendo*.

Hydrogen способен воспроизводить звук на одной из звуковых подсистем – OSS, ALSA, Jack, PortAudio. При первом запуске выбран, кажется (я давно не запускал *Hydrogen* впервые, так что не помню), Jack – поэтому, если вы не слышите звук, запустите предварительно Jack, или переключитесь на ALSA. На мой взгляд, Jack для вывода звука следует использовать лишь в том случае, если вы собираетесь запускать *Hydrogen* одновременно с какой-нибудь другой музыкальной программой.



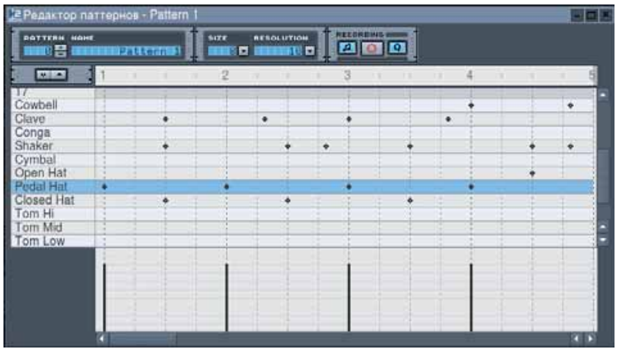
› Рис. 3. *Hydrogen* собственной персоной.

Hydrogen поддерживает MIDI – это значит, что вы можете наигрывать мелодию с MIDI-клавиатуры, а также экспортировать песню в MIDI-файл. Другой формат вывода – WAV. Это наилучший способ работы с *Hydrogen*, поскольку, если вы экспортируете песню в MIDI, то MIDI-файл будет воспроизводиться MIDI-инструментами неизвестно какого качества. А в *Hydrogen* звуки ударных воспроизводятся с помощью наборов загружаемых пользователем звуковых банков.

В составе дистрибутива *Hydrogen* идут два банка: **GMkit** и **TR808EmulationKit**, дополнительные банки можно свободно скачать с сайта продукта – особенно советую вам **UltraAcousticKit**. Оба «стандартных» банка основаны на сэмплах из «железных» драм-машин Roland. **GMkit** – акустические ударные, а **TR808EmulationKit** больше подходит для техно и сходных стилей. Отмечу, что звучание этих банков лучше, нежели стандартные банки той программной драм-машины, которая идет в комплекте с Cubase и Nuendo.

И еще немного о банках. *Hydrogen* работат с сэмплами в форматах WAV и FLAC. Есть встроенный редактор инструментов. В *Hydrogen* вы можете сделать свой собственный банк звуков и экспортировать его во внешний файл, чтобы передать кому-либо (например, на тот же сайт *Hydrogen*).

Интерфейс у *Hydrogen* довольно прост. Композиция состоит из паттернов, в которых вы ставите точки там, где должны играть нужные вам ударные инструменты. В паттерне можно также выставить громкость звучания каждой ноты – при умелом использовании этого приема вы добьетесь относительно «живого» звучания партии ударных, если, конечно же, «живое» звучание вам необходимо. Кроме того, для очеловечивания партии, в микшере предусмотрено три регулятора – Humanize, Timing и Swing, при изменении положения которых *Hydrogen* начнет «ошибаться» при воспроизведении.



› Рис. 4. Окно редактирования паттерна.

Для редактирования паттерна служит окно Редактор паттернов. Чтобы из него воспроизводилась мелодия (а не вся песня целиком), надо на нижней панели управления переключить режим воспроизведения с Song на Pattern. Для удобства включите там же зацикленное воспроизведение.

Если вам надо проиграть всю песню, переключитесь на режим Song. Паттерны воспроизводятся в порядке, заданном в окне Редактор композиций. У вас есть таблица, где в столбик слева записаны названия паттернов, а сверху идет временная шкала. Надо поставить в таблице квадратики в тех местах, где – а вернее, когда – должен звучать соответствующий паттерн. Несколько паттернов могут звучать одновременно.

Наконец, еще одно важное окно – микшер. Кроме настройки – отдельной для каждого канала – громкости и панорамы (положения звука в стереопространстве), в мастер-секции находится кнопка FX, нажав на которую, вы получите секцию из четырех эффектов. Роль эффектов играют плагины LADSPA, которых существует великое множество. Эти плагины можно взять на **http://plugin.org.uk**.



› Рис. 5. Окно микшера.

»

» Чтобы подключить плагин, нажмите кнопку Edit и в открывшемся окне нажмите другую кнопку – «Выберите FX» – а затем уж выбирайте эффект из списка. В микшере в секции эффектов рядом с каждым эффектом есть кнопка BYP (Bypass, обход). Если ее включить, звуковой сигнал будет идти в обход эффекта, то есть эффект будет временно выключен. А от регулятора Return зависит степень возврата звукового сигнала от эффекта. Чем более Return выкручен вправо, тем больше чувствуется влияние эффекта.

Для каждого канала в микшере, под регулятором панорамы, доступно четыре маленьких светло-серых регулятора. Это посылы на эффекты. Они играют примерно ту же роль, что и Return. Чем больше уровень посыла, тем больше инструмент будет обработан эффектом.

Важный практический совет. Как «подружить» *Muse* и *Hydrogen*? *Muse* – многофункциональная цифровая звуковая рабочая станция, о которой мы поговорим в следующей статье. В *Muse* вы можете записывать мелодии MIDI-инструментами, воспроизводить их виртуальными синтезаторами и банками звуков Sound Fonts. Кроме того, в *Muse* можно сводить живой звук, записанный с микрофона или линейного входа. Теперь представим, что мы хотим всю песню сделать в *Muse*, а партию ударных – в *Hydrogen*.

Конечно, можно вначале сделать ударные в *Hydrogen*, затем экспортировать их в WAV и поместить в проект в *Muse*. Но допустим, что нам нужно редактировать ударные по ходу правки всей композиции. Не будем же мы постоянно сохранять из *Hydrogen* в WAV! Нам нужно, чтобы, когда включается воспроизведение песни в *Muse*, начиналось воспроизведении партии ударных в *Hydrogen*, причем с того же момента времени, с которого начал свое воспроизведение *Muse*. Как это сделать? Рецепт таков:

1 Запускаем сервер Jack.

2 Запускаем *Muse* и *Hydrogen*.

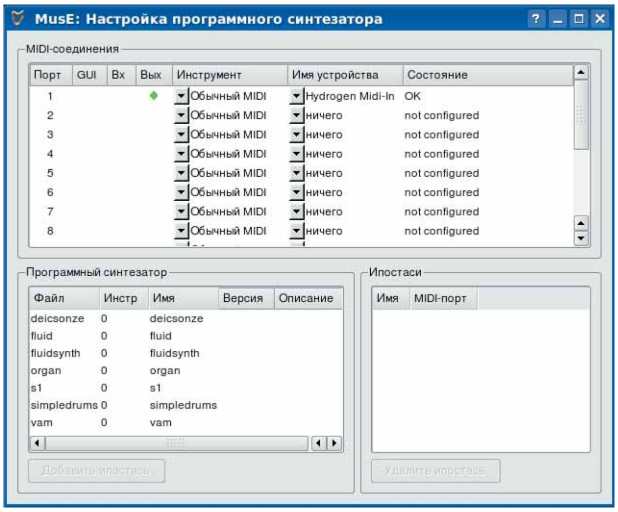
3 В *Hydrogen*, на панели управления, включаем кнопку Jack trans.

4 В *Muse* переходим в Настройки > MIDI-синхронизация, и в разделе «Режим синхр.» включаем «Ведущий». Теперь, если у вас есть хотя бы одна дорожка, то при включении песни на воспроизведение, автоматически будет запущена и композиция в *Hydrogen*.

Усложним задачу. Теперь мы хотим прописать партию ударных в *Muse* и передавать ее на воспроизведение в *Hydrogen*!

5 В *Hydrogen* создаем пустую композицию, загружаем нужный банк звуков.

6 В *Muse*, идем в Настройки > MIDI-порты/Прогр. синтезаторы. Появится вот такое окно:



› **Рис. 6. Настройки *Muse***

Для простоты примера я выбрал выходной MIDI-порт 1. Итак, этому MIDI-порту мы ставим в соответствие *Hydrogen* MIDI-in, выбрав этот пункт в списке Имя устройства. *Hydrogen* MIDI-in – это виртуальный MIDI-порт, предоставляемый *Hydrogen*'ом для приема MIDI-сообщений.

Теперь создаем MIDI-дорожку с ударными (правая кнопка мыши > Добавить дорожку с ударными) и указываем для нее выходной порт *Hydrogen* MIDI-in. Рисуем на MIDI-дорожке произвольный отрезок, выделяем его и открываем редактор ударной партии (Правка > Ударные, Ctrl-D). Редактируем партию. По-прежнему из *Muse*, запускаем песню на воспроизведение. Сходным образом настраивается и MIDI-синхронизации с Rosegarden (еще одна виртуальная рабочая станция).

Завершая рассказ о *Hydrogen*, не могу обойти вниманием некоторые другие его положительные качества. В режиме реального времени, вы можете «играть» на *Hydrogen*'е не только через MIDI-клавиатуру, но и с помощью обычной компьютерной клавиатуры. Для оживления звучания инструментов *Hydrogen* позволяет добавить коэффициент случайного изменения питча (базовой частоты) инструмента. Делается это в редакторе инструментов с помощью регулятора Random pitch. Замечу, что этот способ не следует применять к бас-барабану (kick), потому что в реальной ударной установке в бас-бочку бьют механически с помощью специальной педали с колотушкой. Вы нажимаете ногой на педаль и колотушка лупит по барабану. При этом может изменяться громкость звучания, однако, не его тон. А вот уже при игре на рабочем барабане (он же малый, он же snar) звучание зависит от того, как именно вы ударяете по нему палочкой. Тут Random pitch может пригодиться.

В планах разработчиков *Hydrogen* – переработка интерфейса, устранение окна микшера (с переброской его элементов в другие окна, что, на мой взгляд, не самое лучшее решение), создание программного синтезатора-сэмплера (вместо того, что сейчас доступен из редактора инструмента). Но и то, что уже реализовано в настоящее время, весьма работоспособно. Рабочие качества *Hydrogen* таковы, что его можно использовать как основную драм-машину в студии любительского или профессионального уровня.

LMMS

Web: http://lmms.sourceforge.net

Своим названием *LMMS* подобен знаменитому плейеру XMMS, поэтому человек, впервые услышав про *LMMS*, может решить, что это еще один плейер. На самом же деле «*LMMS*» расшифровывается как «Linux MultiMedia Studio». Это программа для создания музыки, а не только барабанных партий. Я помню *LMMS* с первой версии, когда разработчик сообщал, что-де *LMMS* – это «наш ответ *Cubase*». Однако, на *Cubase* он вовсе не похож, ни внешним видом, ни функциональностью. Скорее, это свободный аналог *Fruity Loops*. *LMMS* – одна из самых красивых музыкальных программ под Linux.



› **Рис. 7. Главное окно *LMMS***

Итак, что же такое *LMMS*? Что он дает вам как музыканту, начинающему или профи? Дорожки, на которых можно записывать музыку. Мелодия пишется либо в пианоролле (эдакая таблица, где вы прямоугольниками рисуете ноты), либо в том, что в русском переводе назва-

но редактором ритма/лейтмотива, а в оригинале – beat+bassline editor, то есть «редактор ударных и басов». Этот последний, практически, повторяет редактор паттернов в драм-машине, то есть каждая нота инструмента/дорожки может находиться лишь в одном состоянии – включена или выключена.

Так же, как каждая квартира начинается с двери, всякая программа начинается с установки. *LMMS* – частый гость в дистрибутивах Linux, куда он входит в уже собранном, бинарном виде. А для установки из исходников вам понадобится *Qt3*. Чтобы собрать *LMMS* на 64-битном дистрибутиве, возможно придется немного подправить скрипт configure, дабы заменить подстроку «$QTDIR/lib» на «$QTDIR/lib64».

Кроме того, вам понадобятся заголовочные файлы библиотек вывода звука, хотя бы одной из следующих: ALSA, Jack, SDL, OSS. Также нужны библиотеки *Ogg Vorbis* (большинство сэмплов из комплекта *LMMS* идут в этом формате), *libsamplerate* и *libsndfile*. Для поддержки плагинов LADSPA исходные тексты нужно сконфигурировать с ключом **--with-ladspa**.

Примечательно, что в исходные тексты *LMMS* (и в пакетные сборки) включены сэмплы и примеры композиций, причем все это не свалено в одну кучу, а кропотливо отсортировано по тематическим папкам, которые доступны во встроенном в *LMMS* браузере. Кроме сэмплера, в *LMMS* включены также пять программных синтезаторов. Пресеты к ним в найдете в том же браузере, в папке Мои предустановки. Звучания этих синтезаторов будет более чем достаточно, если вы пишете музыку в стилях techno, trance или hip-hop, грубо говоря, все то, что можно написать в коммерческом продукте *Fruity Loops*, который, увы, не выпускается в Linux-версии.

Тематические папки с сэмплами и пресетами доступны как вкладка эдакой панели управления, которая находится в левой части главного окна *LMMS*. На вкладках этой панели расположены программные синтезаторы, сэмплы, файловый браузер Мои проекты (отображающий как входящие в состав *LMMS* песни-примеры, так и сохраненные вами проекты) и файловый браузер общего назначения, просто отображающий вашу файловую систему (**Рис. 8**).

Впрочем, не обошлось и без недочетов. Нигде в *LMMS* вы не найдете индикатора уровня выходной громкости. Есть индикатор загрузки процессора, а над ним – осциллятор текущей волновой формы. Индикатора же громкости нет. Общей громкостью можно управлять одним-единственным ползунком, причем громкость в нем измеряется в процентах. Иными словами, правильное управление выходной громкостью сигнала отсутствует. *LMMS* дает возможность включить лимитер, опять-таки общий, который будет гасить перегрузку сигнала, однако никаких опций у этого лимитера нет, присутствует только кнопка в двух состояниях: включено – выключено.

Нет микшера. Громкость дорожки настраивается регулятором рядом с каждой дорожкой. Панорама дорожки доступна только в окне соответствующего дорожке сэмплера или синтезатора, причем в виде двумерного поля с надписью Surround.

Поговорим об инструментах. Любой инструмент в *LMMS*, будь то сэмплер или синтезатор – это плагин. Когда вы хотите сыграть определенную партию сэмплом, то это осуществляется с помощью инструмента-сэмплера Audiofile Processor (**Рис. 9**).

Который, к слову, обладает возможность зацикливать загруженный сэмпл, то есть создавать из него петлю. Кроме того, одним нажатием кнопки устанавливается режим обратного воспроизведения сэмпла, то есть реверс. Прочие инструменты являются синтезаторами разного назначения. Например, Plucked! Stringsynthesis предназначен для эмулирования звучания струнных. Вот как он красиво выглядит (**Рис. 10**).

Синтезатор Organic позволяет создавать не только звуки электро-органа, но и жесткие техно-клавиши – надо только не забывать о наличии кноба Dist (от distortion – искажение) (**Рис. 11**).

Еще один синтезатор – Triple Oscillator – не специализируется ни в чем, поэтому область его применения очень широка (**Рис. 12**).

Каждый синтезатор появляется с стандартном окне с несколькими вкладками и клавиатурой внизу, на которой можно сразу же опробовать инструмент. На первой вкладке (plugin) доступен интерфейс синтезатора. Вторая вкладка – управление фильтрами обработки сигнала, поступающего от синтезатора, а также такими его параметрами, как атака, затухание и тому подобное. Третья вкладка дает возможность включить воспроизведение аккордами и арпеджио – здесь радует огромное количество пресетов. Наконец, на вкладке MIDI можно настроить взаимодействие инструмента с MIDI-устройствами.

Теперь о важном касательно синтезаторов – почти каждый их параметр может быть автоматизирован, для чего служит редактор автоматизации. Чтобы вызвать его, надо нажать правую кнопку мыши



Рис. 8. Вкладка сэмплов. Список «встроенных» звуков.



Рис. 9. Audiofile Processor.



Рис. 10. Plucked! Stringsynthesis.



Рис. 11. Organic.



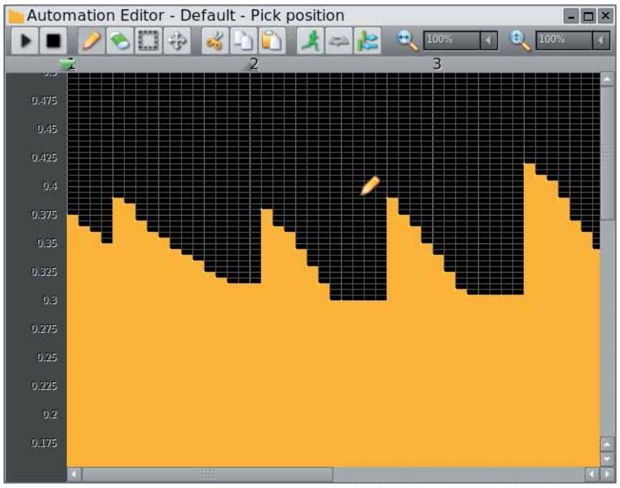
Рис. 12. Triple Oscillator.

Скорая помощь

LMMS способен использовать VST-инструменты, но такой сборки *LMMS* вы, пожалуй, не найдете – по лицензионным соображениям. К счастью, собрать *LMMS* с поддержкой VST можно и самостоятельно. Для этого надо проделать несколько шагов – установить заголовочные файлы **libwine**, установить *Steinberg VST SDK* (версий 2.3 либо 2.4 – **http://www.steinberg.de/331+M52087573ab0.html)**, скопировать файлы **aeffect.h** и **aeffectx.h** в каталог «**lmms-0.2.x/include**» и наконец, запустить настройку исходных текстов:

```
./configure --with-vst
```

на элементе управления и выбрать в контекстном меню пункт Open in automation editor (в момент написания этих строк он еще не был переведен на русский).



› **Рис. 13. Automation editor.**

В этом редакторе вы можете карандашиком рисовать изменения выбранного элемента управления на временной шкале. Таким образом можно динамически изменять, допустим, громкость инструмента на протяжении нужного вам отрезка времени, либо влиять на звучание синтезируемого звука. Наличие автоматизации такого уровня говорит о высоком классе *LMMS* среди прочего звукового ПО, будь то для Linux, Windows или MacOS.

LMMS вообще производит хорошее впечатление уже с самого начала. Редкая программа в первых версиях имеет такой цельный и, я бы сказал, модный графический интерфейс. Разработчикам пришлось для него много рисовать и программировать, ведь в *LMMS* используется большое количество нестандартных виджетов, которых в *Qt* просто нет – взять хотя бы фортепианную клавиатуру.

Интерфейс *LMMS* на первый взгляд кажется сыроватым – вероятно, из-за большого размера кнопок и некоторых других элементов управления, но потом взгляд привыкает. Кроме того, нельзя отрицать простоту использования *LMMS* – если, конечно, вы сталкивались ранее с подобными программами, той же *Fruity Loops*. Дружественность интерфейса *LMMS* играет очень важную роль – программу можно освоить методом «тыка». Научный «тык» необходим, поскольку за два года существования *LMMS* никто не удосужился написать к нему документацию. Некоторые элементы интерфейса снабжены контекстной подсказкой, но она не может заменить полноценное руководство. На сайте *LMMS* есть Wiki с информацией по *LMMS*, однако, она тоже не может претендовать на звание документации. Плохо освещены даже основные рабочие характеристики *LMMS*. Непонятно, какова разрядность внутреннего микширования – 16, 32, 64 бита? Как действует встроенный лимитер? Что за кнопка «высокое качество», на что именно она влияет? Где доступ к плагинам LADSPA, если я собрал *LMMS* с поддержкой этой технологии?

В *LMMS* есть импорт песен от *Fruity Loops*. В *Wiki* такая возможность заявлена как планируемая. Я нашел у себя в закромах собственные, древние композиции, написанные во *Fruity Loops*, и попробовал их импортировать. В самом деле, импорт некоторым образом работает – загружаются дорожки с нотами, но все дорожки (и с сэмплами, и синтезаторные) трактуются *LMMS* как дорожки для инструмента Audio File Processor. Кроме того, после импорта вам придется заново назначать инструментам сэмплы, но это уж не вина *LMMS* – ведь у *Fruity Loops* свои инструменты, сэмплы. А в файлах *Fruity Loops* хранятся, грубо говоря, только мелодии.

Еще в *LMMS* есть импорт MIDI, он тоже работает – сходным образом с импортом *Fruity Loops*, то есть после импорта вы получаете дорожки с партиями, а уж забота по распределению для них инструментов ложится на ваши плечи.

Впечатления от *LMMS* следующие. Разработчики мало внимания уделяют общему, однако много – частному. Частное – допустим, есть окно «Заметки» с текстовым редактором, куда вы можете добавлять какие-то свои заметки к проекту. При этом текст не простой, а с оформлением – доступно меню для его форматирования и раскраски разными цветами. Это очень здорово, но почему такая, в принципе, второстепенная для музыки штука реализована, а индикатор громкости – нет? Я понимаю, что при микшировании *LMMS* все равно рубит громкость на нуле децибел, но мне нужно видеть уровни громкости и иметь возможность наглядно ими управлять. Однако должен признать, что программ такого класса и возможностей, как *LMMS*, довольно мало. Более того, под Windows у *LMMS* есть только коммерческие аналоги – *FruityLoops* и *Orion*.

Что до аналогов *Hydrogen* под Windows, то я давно не видел в этой системе драм-машин, которые не сделаны в виде плагинов. Помню, Steinberg одно время выпускала «отдельную» программную драм-машину *BBox* со славным набором сэмплов. Вот *Hydrogen* по ощущению работы больше всего похожа на этот *BoomBox*, хотя возможностей у *Hydrogen* больше.

Остается прибавить, что следуя добрым традициям Linux, *Hydrogen* и *LMMS* распространяются под лицензией GPL, и денег за свои продукты разработчики не берут. И в какой-нибудь другой стране, где пиратства, условно говоря, нет, музыкант-любитель вполне может позволить себе работать под Linux с хорошим набором софта, не тратя на это ни одной заграничной копейки (кроме как на скачивание из Сети). LXF

Сэмпл или образец?

Пользуясь случаем, немного покритикую русский перевод LMMS. Вкладку «Мои образцы», где находятся сэмплы, можно было бы перевести как «Мои сэмплы» – так понятнее, ибо слово «сэмпл» уже прижилось в околомузыкальной среде, и никто не говорит про сэмплы – «образцы».

Я сам противник вливания в язык иностранных слов, но слово «сэмпл» так же привычно, как, например, слово «джинсы». И хотя «jean» переводится просто – «плотная хлопчатобумажная ткань», мы говорим «надел джинсы», а не «надел плотную хлопчатобумажную ткань». На худой конец, можно использовать вместо «сэмплы» слово «звуки», однако не «образцы». Образцы выставляют в витрине или привозят на выставку.

» Через месяц Нас ждет тяжелая артиллерия - цифровые студии звукозаписи.



Учебники

Наши эксперты помогут вам с любым приложением Linux



ЕВГЕНИЙ БАЛДИН
Интеллигент в четвёртом поколении и русский как минимум в седьмом.

Пора заявлять о своих правах

«Carpe Jugulum.
Хватай за горло!»
Терри Пратчетт

Недавно я зашёл на один сайт с целью оптимизации своего времени по управлению своими, увы, пока немногочисленными финансовыми операциями и вот что увидел в вопросах для пользователей:

В: Что нужно для работы в системе Интернет-банкинг *******.**?

О: Для работы в системе Интернет-банкинг *******.** вам потребуется лишь компьютер, подключённый к сети Интернет.

Я обрадовался, но призадумался, а правда ли это и написал письмо: «У меня есть компьютер, на нём стоит дистрибутив GNU/Linux Debian и я использую браузер Mozilla, к Интернету я подключён и имею счёт в банке, который Вы обслуживаете. Могу ли я воспользоваться Вашим сервисом? И что мне нужно для этого сделать?»

И что же мне ответили? «К сожалению ****** будет работать под ОС Win 98/2k/XP и с браузером IE версии минимум 5.5. Связано это с использованием ActiveX компонента, поддержка которого в полном объёме реализована только в IE».

Налицо явное пренебрежение. С другой стороны, понятно, почему это происходит – сами разработчики и заказчики вполне могут и не знать, что есть что-то, кроме Windows. Я думаю, сейчас самое время рассказать им, что они неправы. И не надо стесняться это делать.

Попробуйте найти подобный упомянутому выше сервис и написать запрос по поводу его работоспособности в GNU/Linux. Одно из основных свойств этого «домашнего задания» – оно не обязательно к выполнению, но когда-то начинать придётся. Мир должен потихонечку узнавать про свободные системы, хотя бы через письма потенциальных клиентов.

P.S. Желающие узнать о работоспособности конкретного описанного мной сервиса могут получить его адрес у меня по e-mail.

E.M.Baldin@inp.nsk.su

В этом выпуске...

Совет месяца: Перенаправление



Даже если вы новичок в Linux, вы наверняка использовали перенаправление при работе в командной строке. Для обмена данными с выполняемыми командами используются символы > и <. Чаще всего их применяют, чтобы сохранить вывод команды в файле. Например, набрав dmesg > local.log, вы запишите содержимое кольцевого буфера ядра (иными словами, вывод *dmesg*) в файл **local.log**.

> local.log предписывает перезаписать указанный файл, если он уже существует. Использовав >> вместо >, можно дописать вывод *dmesg* в **local.log**. Символ < подает содержимое файла-аргумента на вход команды, например, *grep*. Так, команда grep USB < local.log осуществит поиск строки «USB» в файле **local.log**.

Перенаправление работает, поскольку любое приложение Linux осуществляет ввод/вывод посредством трех стандартных «файловых дескрипторов» – стандартного ввода (stdin), вывода (stdout) и потока ошибок (stderr). Обычно вы этого не замечаете, поскольку по умолчанию этим дескрипторам соответствуют экран и клавиатура. Символы < и >, использованные в предыдущих примерах, относятся к stdin и stdout. Как же тогда обратиться к стандартному потоку ошибок? Для этого необходимо добавить перед символом > цифру 2 – номер файлового дескриптора, отвечающего stderr. Стандартному вводу соответствует значение 0, а выводу – 1. Это полезно знать, поскольку таким образом можно отфильтровать сообщения об ошибках от полезного вывода программы.

Например, рассмотрим команду *find*. Она часто жалуется на проблемы с правами доступа. Заставить ее замолчать можно, перенаправив стандартный поток ошибок в **/dev/null** – специальное устройство, поглощающее все, что было в него послано:

```
find / -name *.jpg 2>/dev/null
```

APT: Работа

Вам нужна новая программа, причем немедленно!? **Энди Ченнел** развеет миф о том, что «приложения трудно устанавливать», раз и навсегда.





Наш эксперт

Энди Ченнел
Энди делает свои первые шаги в Linux уже шесть лет, а технологиями интересуется еще со времен Dragon 32.

Средний дистрибутив Linux отнюдь не беден приложениями. Некоторые из них содержат более 2000 пакетов: стандартные офисные приложения, web-редакторы, игры, обучающее ПО и прорва всего прочего. А так как разработчики Linux-приложений склоняются к принципу «релизы раньше, релизы чаще», то пакеты, составляющие дистрибутив, часто устаревают еще до того, как CD/DVD выйдут с фабрики. К счастью, пользователи Linux могут устанавливать или «освежать» большую часть приложений при помощи единого менеджера пакетов, и все популярные дистрибутивы предусматривают автоматическое обновление системы – в отличие от обновлений и установок, с которыми маются пользователи Windows.

На этом уроке мы рассмотрим пару методов управления вашими приложениями, доступных в Ubuntu: начнем с автоматической и ручной систем обновления, затем перейдем к установке приложений из командной строки и в графическом режиме из репозиториев Ubuntu, а завершим установкой файла Deb, найденного в Интернете, чтобы проверить, как Ubuntu отслеживает зависимости.

Что такое зависимости? Большинство приложений Linux для выполнения неких действий нуждаются (зависят) в других приложениях. Например, *Gimp* – графический редактор – зависит от пакета с названием *PyGTK*, который нужен ему, чтобы выполнять Python-скрипты в некоторых своих фильтрах, и для правильной работы *Gimp* сначала нужно установить этот пакет. Кроме того, *PyGTK* имеет свои требования по наборам зависимостей (собственно Python), и их необходимо удовлетворить перед его установкой. Процесс удовлетворения таких требований вручную приводит к жуткому термину «ад зависимостей», когда пользователь ходит по кругу, отыскивая все пакеты для удовлетворения всех зависимостей. Хотя ад зависимостей теперь практически в прошлом, легенды о нем способны отпугнуть новых пользователей от перехода на Linux.

Автоматическое обновление

Мы будем работать в Ubuntu, но те же действия и приложения доступны в любом основанном на Debian дистрибутиве (типа Mepis, Knoppix или Linspire). Но имейте в виду: современные дистрибутивы полагаются на постоянную связь с Интернетом, и если у вас ее нет, то многим из описанных шагов будет предшествовать команда ‘подключитесь к Интернет’ [и выберите канал пошире да подешевле, – прим. ред.].

Первым делом надо убедиться, что сам дистрибутив полностью актуален. В Ubuntu имеется апплет, автоматически запускающийся при загрузке, который подключается к серверу проекта для проверки актуальности дистрибутива. Если есть какие-либо обновления, то в области уведомлений, справа и вверху экрана Ubuntu, появляется небольшая иконка (прямоугольник со звездочкой), а под ней – всплывающее окно с сообщением, что обновления доступны. В этой ситуации вы можете запустить менеджер обновлений, щелкнув на иконке, и следовать указаниям. Иногда обновления бывают достаточно тривиальны, но могут и включать важные исправления безопасности – если вы используете соединение через телефонную линию, то можете захотеть прочитать описания заплаток и установить очередность их получения. Используйте маленькую раскрывающуюся стрелку с подписью Changes And Description Of The Update (Изменения и Описание Обновлений), чтобы узнать, что поменяется. Следует отметить, что Ubuntu обычно показывает важные обновления безопасности перед обновлением приложений, так что вам, возможно, придется пройти эту процедуру дважды.



› **Командная строка не обязана пугать. Запуск CLI-приложений не сложнее нажатия кнопки.**

Месяц назад Сканирование изображений и текста в Linux при помощи QuiteInsane.

с пакетами

Пакетная терминология

» APT Это *Advanced Package Tool* (Продвинутый инструмент для пакетов), разработанный и поддерживаемый Debian. Когда вы устанавливаете приложения, он пытается взять на себя всю заботу о зависимостях.

» Autopackage Попытка упростить установку приложений после предварительной загрузки и небольшой настройки. Любые пакеты Autopackage должны устанавливаться корректно сразу же после двойного щелчка. Список доступных пакетов можно найти на **http://autopackage.org**.

» Командная строка Хотя в учебнике мы стремимся обойтись графическими инструментами, иногда следует воспользоваться командной строкой: это способ подачи команд компьютеру при помощи строк текста, а не нажатий кнопок мыши или пунктов меню. Командная строка доступна при использовании такого приложения, как *Terminal*, *Console* или *Konsole*, которые обычно спрятаны в меню Accessories.

» .deb Родным для Debian является бинарный формат пакетов Deb. Если вы ищите конкретное приложение в поисковике, добавляйте к искомому тексту '.deb', чтобы быть уверенными, что найдете правильный пакет.

» Зависимости Некоторые приложения для корректной работы нуждаются в других программах или библиотеках, поэтому они должны быть установлены до установки самого приложения. Раньше это было сложно, но *APT* действительно хорошо справляется с зависимостями.

» Репозиторий *APT* использует систему репозиториев – on-line хранилищ, заполненных программами. Вы можете добавлять в вашу систему новые репозитории: например, можно обновить старую версию Ubuntu до новейшего издания, используя самый свежий репозиторий.

» RPM Debian имеет *APT*, а Red Hat, Mandriva и другие используют систему установки, основанную на Red Hat Package Manager (менеджер пакетов Red Hat). Пакеты, использующие эту систему, имеют расширение **.rpm**.

» Исходные тексты Некоторые пакеты, особенно новейшие версии, распространяются в виде исходных текстов, что означает необходимость сборки и установки их пользователем. Звучит кошмарно, но не так все плохо. Мы дадим краткое введение в установку из исходных текстов в следующем номере.

Если опции уведомления нет, можно запустить процедуру вручную, используя System > Administration > Update Manager (Система > Администрирование > Менеджер обновлений). Затем менеджер обновлений просмотрит сервер в поисках новых версий каждого пакета на вашем ПК и, если таковые имеются, представит их для установки. Вы можете отменить обновление отдельных приложений, убрав «галочку» рядом с их именем.

CLI для начинающих

Syn*APT*ic – прекрасная оболочка для *APT* (она преобразует выбранные мышью опции в команды, которые затем отправляются системе), но иногда бывает разумно воспользоваться командной строкой напрямую. Открыть терминал и ввести несколько команд оказывается быстрее, чем шаркать мышью. Поэтому первое, что нам необходимо для работы – терминал, или консоль. Обычно их один или два, и они располагаются в пункте меню Accessories. Используйте тот, что вам удобнее.

Терминал (консоль) – это текстовый экран, где (обычно в левом верхнем углу) отображается небольшой текст, соответствующий текущему пользователю (то есть andy) и текущему имени машины (то есть V2-Ubuntu), и сопровождается это парой символов, сообщающих, что текст можно вводить – обычно доллар ($) или решетка (#).

```
andy@V2-Ubuntu:~$
```

Затем идет мерцающий курсор, показывающий, где набирать команду. Наберите её, нажмите клавишу Enter, и команда выполнится. Вот эта –

```
firefox
```

запустит web-браузер *Firefox* (если он установлен). Можно не только запустить программу, но и «передать ей аргументы» – проще говоря, указать приложению выполнить что-то кроме простого запуска.

Команда

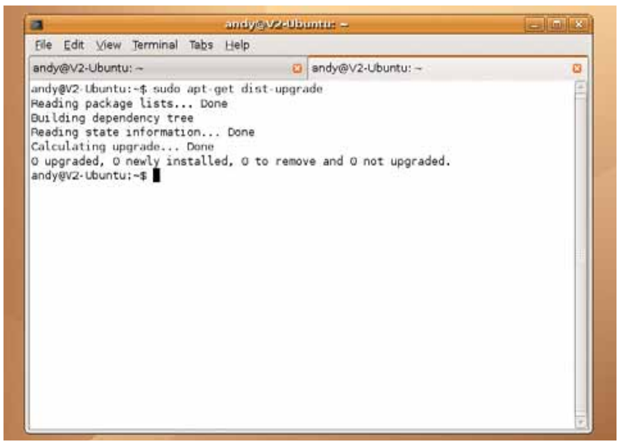
```
firefox "http://www.linuxformat.ru"
```

велит компьютеру запустить *Firefox*, а затем открыть web-сайт **Linux Format**. А при чем тут установка приложений? Ну, *APT* – тоже приложение (команда для его запуска: *APT*-get), и мы можем указать для него параметры, как и для *Firefox*. Вот пример обычной команды на установку:

```
apt-get install inkscape
```

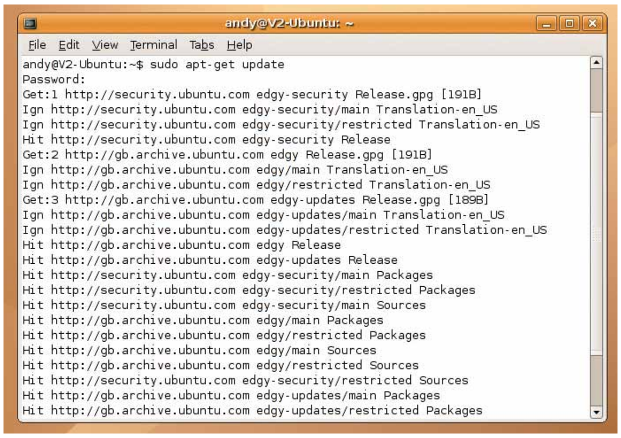
В общем, это значит «запустить приложение *APT*-get и загрузить и установить пакет *Inkscape»*. Но если вы просто введете эту команду в строке, она не сработает. Почему? Потому что для установки приложений в системе Linux необходимы привилегии суперпользователя (root). Если вы привыкли к другим операционным системам с меньшей защищенностью, то это может показаться излишним, но это также означает, что вирусы, трояны и руткиты не смогут установить себя в вашей системе без вашего ведома. Чтобы стать root'ом в Ubuntu, добавьте sudo в начало команды apt-get и на запрос введите свой пароль. (Для выполнения этого в других дистрибутивах, вводите su, на запрос введите пароль root, затем введите apt-get install inkscape.)

Приложение просканирует все доступные репозитории в поисках *Inkscape*. Найдя его, оно проверит зависимости пакета и вновь начнет сканирование на предмет наличия зависимостей. Затем создаст список зависимостей в правильном порядке, загрузит и установит их до загрузки основного пакета, установит его и (скорее всего) добавит пункт меню в соответствующем месте. Впечатляющая работа для четырех коротких словечек. »



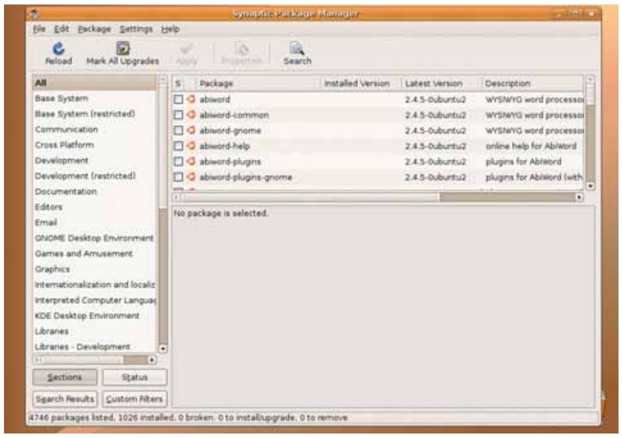
› *APT* – легкий способ обновить тысячи программных пакетов.

Шаг за шагом: Установка приложений в Synaptic



1 Получите Synaptic

Ubuntu использует *Synaptic* в качестве менеджера пакетов по умолчанию, но если у вас его нет, наберите **sudo apt-get install synaptic**, введите пароль и нажмите Enter.



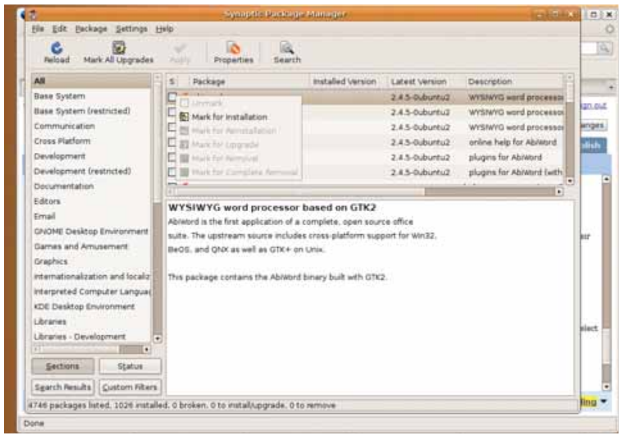
2 Найдите его в командной строке

После установки *Synaptic* должен быть доступен в системном или меню настройки, но если вы не можете найти его, откройте терминал еще раз и введите **sudo synaptic**.



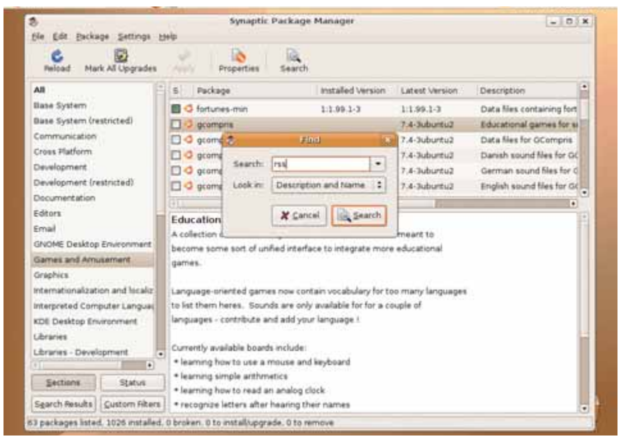
3 Интерфейс APT

Слева находится список категорий – выбор любого пункта здесь приведет к отображению в правой верхней панели всех доступных приложений данной категории. Выберите опцию отображения описания приложения в нижней правой панели.



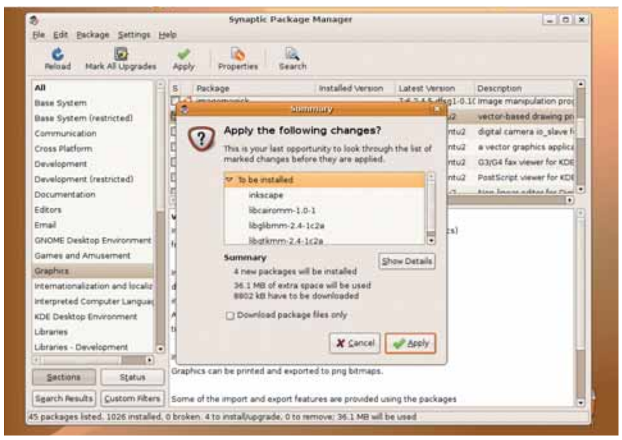
4 Выберите опции

Щелкните на радиокнопке рядом с именем приложения для отображения его опций, затем выберите, следует ли его установить, удалить, и т.д. *APT* сам справится с зависимостями, установит приложение и произведет изменения в меню рабочего стола.



5 Ищите в репозитории

Если вы не хотите просматривать тысячи пакетов, нажмите кнопку поиска на панели инструментов *Synaptic*. Откроется небольшое окно поиска, где вы можете ввести запрос. Доступен поиск по названию, описанию и многому другому.



6 Получите новые приложения!

Выбрав пакеты для установки, удаления или обновления, нажмите кнопку Применить (Apply) для запуска процесса, и обновление или установка начнется. Ваши приложения должны появиться, как по волшебству, под правым заголовком в вашем меню!

» Если установка одного приложения четырьмя словами хороша, вообразите, сколь удивительно набрать apt-get upgrade и обновить каждый пакет – ну, из тех, что доступны на серверах Debian/Ubuntu – до свежайшей имеющейся версии. Вы можете даже обновить каждый элемент вашей системы – от ядра до KDE – до последней версии, просто набрав:

```
apt-get update
apt-get dist-upgrade
```

Первая команда гарантирует полную синхронизацию вашей локальной базы данных *APT* (которая содержит список доступных приложений) с удаленным сервером, содержащим программы, а вторая начинает процесс обновления вашего компьютера.

Inkscape, как бесплатное приложение, содержится в стандартных репозиториях Ubuntu. Может случиться, что вам потребуется установить нечто не доступное в Ubuntu; тогда надо будет добавить другой репозиторий, содержащий требуемую программу, и использовать его или в *SynAPTic*, или в командной строке. Ради простоты мы ограничимся *SynAPTic* (если вы хотите добавить репозиторий вручную, необходимо подправить – от имени root – файл **/etc/APT/sources.list**), так что начните с его запуска.

Добавление репозитория

Я хочу установить *Skype*, но быстрый поиск в главном репозитории Ubuntu показывает, что такой программы нет – это, в конце концов, несвободная программа, и потому не соответствует строгой линии свободного ПО, лежащей в основе таких дистрибутивов, как Debian и Ubuntu. Однако разработчики *Skype* поддерживают свой собственный, отдельный от Debian репозиторий, доступный пользователям, и мы можем добавить его в список Syn*APT*ic, нажав последовательно Settings > Repositories (Настройки > Репозитории). В строке Software Sources (Источники приложений), выберите вкладку Third Party (Сторонние), затем нажмите кнопку Add (Добавить). Введите следующее в предоставленном пространстве:

```
deb http://download.skype.com/linux/repos/debian/
stable non-free
```

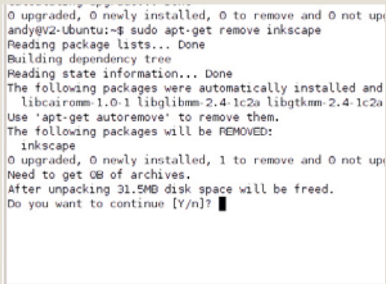
а затем нажмите кнопку Add Source (Добавить источник). Наконец, нажмите Close (закрыть). *SynAPTic* теперь должен сообщить вам, что репозитории изменились и что вам надо нажать кнопку Reload (Обновить) для обновления вашей локальной базы данных. Идея хорошая; вы так и сделайте, а затем поищите *Skype* вновь. На сей раз результат должен разительно отличаться (то есть приложение должно найтись), и более того, когда появится новый релиз *Skype* или он обновится, вы будете уведомлены о том, что доступна новая версия и пакет может быть обновлен без «танцев с бубнами».

Что дарует APT...

Это уже не про установку: APT также может быть инструментом удаления. И синтаксис предельно прост.

```
apt-get remove inkscape
```

Можно выполнять и другие операции. Например, apt-get install firefox -s выполнит «холостой» запуск установки, чтобы увидеть, все ли пройдет гладко, а apt-get check просмотрит приложения в системе, чтобы определить, нет ли оборванных зависимостей. Очень важно время от времени выполнять последнее, это часть стандартной процедуры обслуживания.



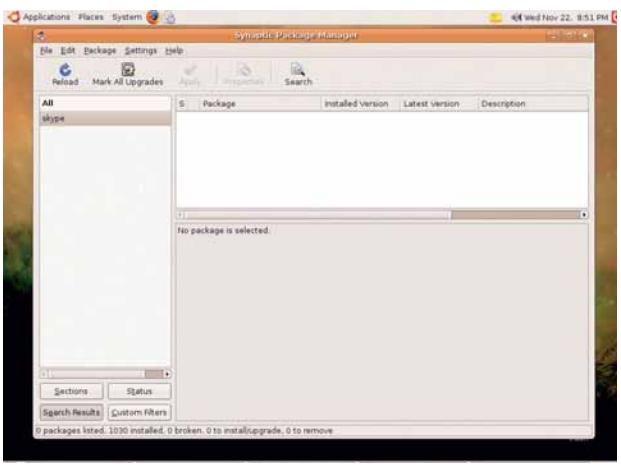
› APT не удаляет приложения без вашего подтверждения.

Установка «найденного» приложения

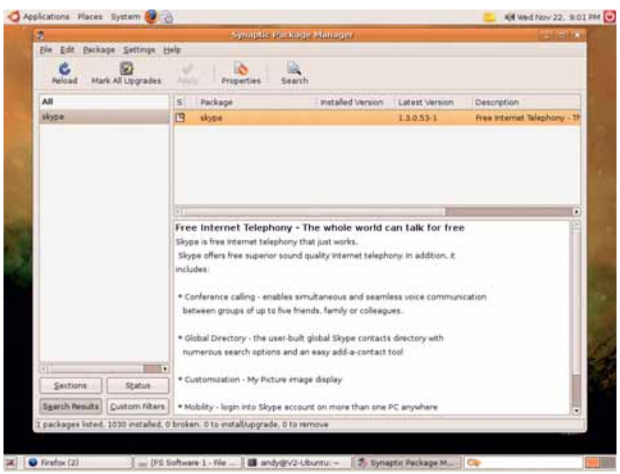
Некоторых приложений просто нет в доступных репозиториях Debian/Ubuntu: они могут быть слишком новыми или (это менее вероятно) слишком специфичными, чтобы быть включенными в них. Теперь небольшая оговорка: *APT* берет на себя заботу о поддержании системы в порядке – тем он и хорош. Установка отдельных пакетов не из стандартных репозиториев может, в некоторых случаях, внести небольшой беспорядок в вашу систему. Имея это в виду, продолжим. Команда для установки нетипичного Deb-файла –

```
dpkg -i packagename.deb
```

Как и до этого, первая часть, dpkg, это имя утилиты; -i – команда установки; а packagename.deb – имя устанавливаемого приложения. Вы можете также удалить пакет при помощи -r, но сохранить файлы конфигурации (они пригодятся при переустановке) или удалить и их при помощи -p, чтобы вычистить все. Однако учтите, что dpkg следует рассматривать как последнее прибежище, потому что с ним возможны проблемы: он устанавливает только указанные вами двоичные файлы, а удовлетворение зависимостей ложится на ваши плечи. Зато *APT* будет поддерживать вашу безопасность и защищенность, а вашу машину – в актуальном состоянии, и обеспечивать получение всех новейших приложений автоматически и бесплатно, при помощи всего лишь пары команд. Разве это не здорово? LXF



› Если вам нужен пакет Ebay VoIP, то вам не повезло...

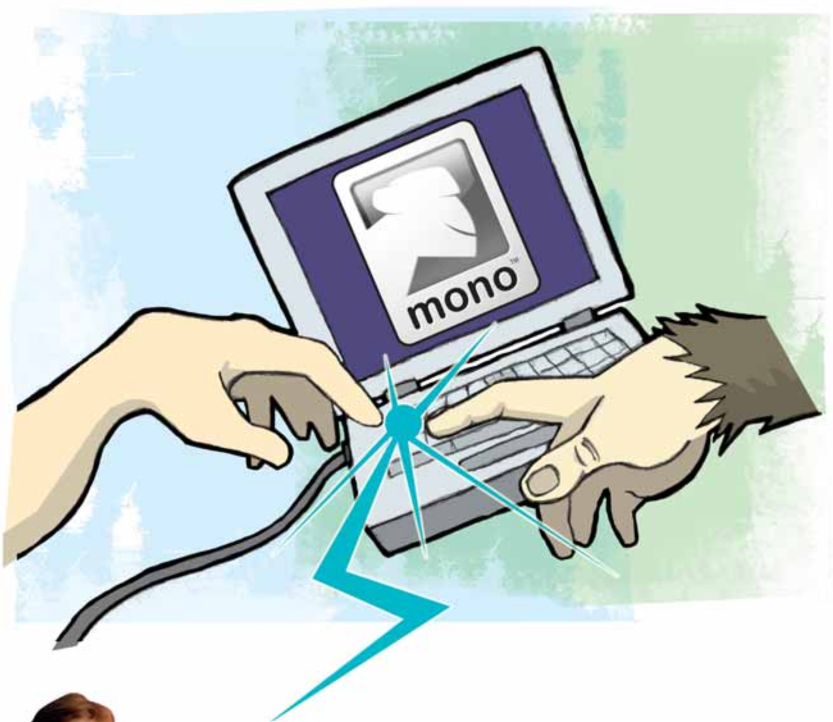


› ...если вы не добавили в *Synaptic* репозиторий *Skype*.

» Через месяц Мы рассмотрим другие возможности для установки программ, включая RPM.

Mono: Работаем

Подогрев ваш интерес пространствами имен и объектно-ориентированным программированием, **Пол Хадсон** покажет, как пишется полезный код....

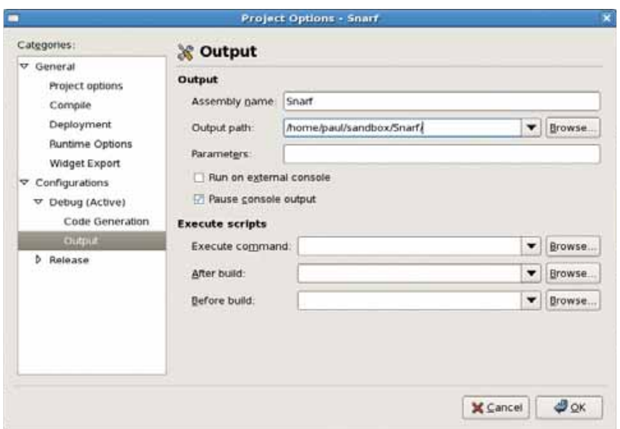




Наш эксперт

Пол Хадсон полагает, что Mono – лучшая вещь со времен мультфильма *Pinky and the Brain*, и сейчас поддерживает два проекта на основе Mono на SourceForge.

Вы когда-нибудь смотрели мультсериал *Thundercats* [ГромоКоты, – прим. пер.]? В детстве я считал его гениальным: вы действительно могли ощущать «колдовство и рык звериный», глядя в телевизор, благодаря крутой анимации, симпатичным персонажам и замечательным сценариям. Сейчас, однако, я понимаю, что он был не лишен шаблонности. Среди ГромоКотов были такие персонажи, как Лева-О, Тигра и Пантро, и они боролись против обезьяноподобного индивида по имени Обезмен, шакаловидного Шаклмена и коршунообразного Коршмена. Их родная страна называлась Громада. Машина Пантро называлась Громокар. Уловили закономерность?



› **Велите *MonoDevelop* компилировать программы в каталог вашего проекта, чтобы программа могла читать кэш файлов.**

Такая предсказуемость может показаться чересчур лобовой, зато дети легко улавливают, что происходит, и легко это запоминают, чтобы пересказать сюжет друзьям. Теперь, повзрослев, я уяснил две вещи. Во-первых, я не стану космонавтом. Я бы и пошел, но вряд ли туда возьмут «очкарика». Во вторых, лучший способ что-то выучить – сделать это запоминающимся.

К примеру, я использую PHP уже много лет и никак не могу запомнить, принимает ли функция strpos() (поиск вхождения одной строки в другую) параметры как $иголка, $стог_сена или как $стог_сена, $иголка.

Проблема PHP в том, что функция strpos() принимает параметры как $стог_сена, $иголка, в то время как функция in_array() (она отвечает на вопрос, встречается ли элемент в массиве) принимает параметры как $иголка, $стог_сена. Вдобавок strpos() пишется в одно слово, а str_replace() содержит разделяющий две части знак подчеркивания. По своей природе, PHP не очень запоминающийся язык программирования. Программистам PHP не быть ГромоКотами.

К счастью, вы не учитесь программировать на PHP, а работаете на платформе Mono с C#, а C# – язык более новый и менее набитый ляпсусами, чем PHP – не имеет таких проблем. Собственно говоря, некоторые части C# так прямолинейны, что вы можете вставлять имена методов по догадке. На нашем втором уроке мы рассмотрим управление файлами. Вы напишете программу, которая пройдется по файловой системе и составит указатель по всем имеющимся файлам, а вы потом сможете его распечатать. Не волнуйтесь, если это звучит сложно – C#, Mono и .NET сделают всю трудную работу за вас.

Ощутите колдовство

Начнем с простейшего чтения и записи файлов. Вообще-то точнее будет сказать – простейшего ввода-вывода: в терминах программирования, «писать» означает «вывести» или «отобразить» (то, что делает команда write). Запустите *MonoDevelop* (который мы установили в прошлый раз) и создайте консольный проект C# (File > New Project > C# > Console Project). Это не временный проект, поэтому дайте ему такое имя, чтобы не стыдно было выложить его на SourceForge. Я выбрал имя Snarf, потому что оно означает «брать» (эта программа будет читать содержимое кучи файлов), а также хорошо сочетается с темой *ГромоКотов*.

Нам понадобится кое-что из новой функциональности .NET 2.0, но в большинстве версий *MonoDevelop* по умолчанию используется .NET 1.1. Чтобы поправить дело, нажмите Project > Options, затем выберите Runtime Options из списка Categories в появившемся окне. Справа помещен выбор среды исполнения, варианты: 1.1 и 2.0, вот и выберите 2.0. Пока вы еще в этом окне, откройте категорию Configurations > Debug, выберите Output, затем измените Output Path, удалив раздел **/bin/Debug**. Теперь *MonoDevelop* будет сохранять исполняемый файл а корневом каталоге вашего проекта. Нажмите OK, чтобы запомнить изменения.

» **Месяц назад** Для первого знакомства с Mono мы написали программу Hello World.



с файлами

И чтение, и запись файлов осуществляются с помощью библиотеки *System.IO* (сокращение от Input/Output – ввод/вывод, т.е. чтение и запись), поэтому нужно поместить 'using System.IO' вверху вашего главного файла проекта. Измените class MainClass на class Snarf и удалите строку Console.WriteLine(), оставив метод Main() пустым.

Первое, что мы сделаем – считаем содержимое одного файла. Содержимое файлов – по крайней мере, тех, что нас интересуют – простой текст, а значит, его можно легко сохранить как данные строкового типа. Создайте файл с именем **myfile.txt** в корневом каталоге вашего проекта (там, где *MonoDevelop* будет сохранять вашу программу), и введите любой текст.

Встает вопрос: как прочесть содержимое файла в строку? А попробуйте так:

```
string myfile = File.ReadAllText("myfile.txt");
```

Вот и все – все, что требуется, чтобы прочитать файл в строку средствами Mono. Для ее вывода можете использовать метод Console.Write(), рассмотренный на прошлом уроке:

```
Console.Write(myfile);
```

Нажмите F5, программа скомпилируется и запустится, и вы должны увидеть содержимое вашего файла в панели Application Output [Вывод приложения] внизу *MonoDevelop*.

Алло, оператор?

Пойдем дальше: заставим нашу программу записывать изменения обратно в файл. Добавьте следующие строчки после вызова Console.Write():

```
myfile += "\nМои крылья как щит... ой, это из другого мультика.";
File.WriteAllText("myfile.txt", myfile);
```

В этом коде += является оператором: это такой символ, который выполняет операцию (лихо закручено, а?). Из школьной арифметики вы знаете операторы +, *, -, /, они выполняются над двумя числами, по-научному – операндами. А оператор = берет значение одного операнда и присваивает его другому. Так, выражение **a=10** означает, что переменная a принимает значение **10**.

Сейчас ваш труд окупится. Если a равно **10**, то как прибавить к нему еще 10? Ага-ага…

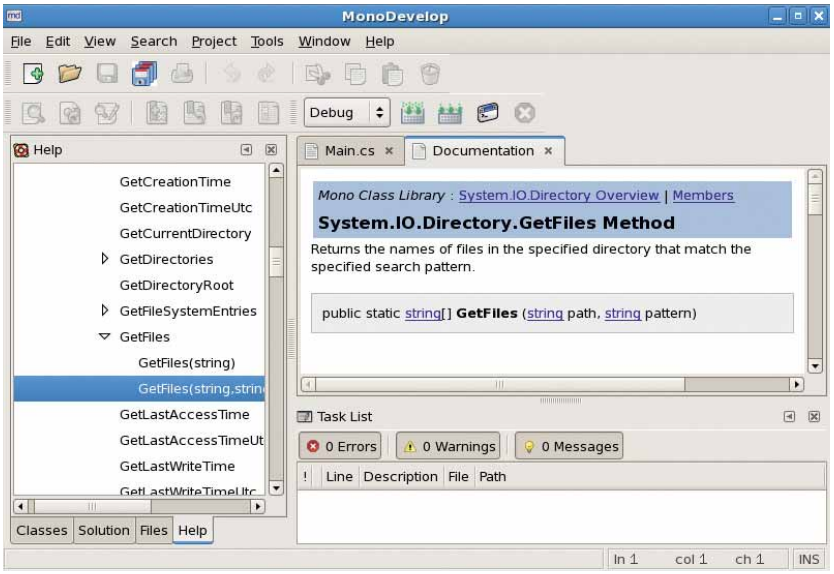
```
a = a + 10;
```

Работает! Согласно принципу «бритвы Оккама» («при прочих равных условиях, лучшим объяснением будет простейшее»), этот код вообще является наилучшим. Принцип, однако, имеет малоизвестное добавление, под названием «поправка Оккама»: в простейшем решении непременно кроются недостатки. В нашем случае, недостаток тот, что для набора строки требуется 11 нажатий клавиш, а C# позволяет сделать то же самое за 8:

```
a += 10;
```

Здесь применен оператор +=, дитя любви операторов + (сложения) и = (присваивания): он прибавляет то, что справа от него, к тому, что слева. Класс! Теперь, с новообретенными знаниями, вы поймете, что код нашей программы добавляет строку (**Мои крылья как щит…** – ну, вы знаете, откуда это) к уже существующей строке. Языку C# хватает ума различить, когда += используется над числами (для сложения двух чисел), а когда – над строками (для конкатенации двух строк). Мы начинаем новую строку с \n, это информирует Mono о необходимости добавить символ новой строки в существующем файле.

Метод WriteAllText() относится к методу ReadAllText() как Wilykit к Wilykat: дайте ему имя файла в качестве первого параметра и текст для записи в качестве второго, а он выполнит всю работу.



› **Левая панель *MonoDevelop* показывает файл решения (по умолчанию) или онлайн-справку. Учтите: некоторые страницы помощи устарели!**

Пора вдарить по газам Громокара и ввести конструкции посерьезнее: рассмотрим условные выражения и циклы. Условные выражения нужны, чтобы выполнять действия только при выполнении (истинности) определенного нами условия. Если условное выражение ложно – небо не голубое, возраст пользователя не 26, или что мы там проверяем – то код исполняться не будет. Например:

```
string Name = "Cheetara";
if (Name == "Snarf") {
  Console.WriteLine("Snarf snarf!");
}
```

Данный код ничего не напечатает: хотя переменная Name существует, ее содержимое не **Snarf**, а **Cheetara**. Заметим, что == просто еще один оператор, означающий «равняется». Он отличается от оператора присваивания = (*см. врезку «Когда = ведет к ошибке»*).

Знакомство с циклом

Циклы позволяют выполнять определенный блок кода несколько раз. Например:

»

Когда = ведет к ошибке

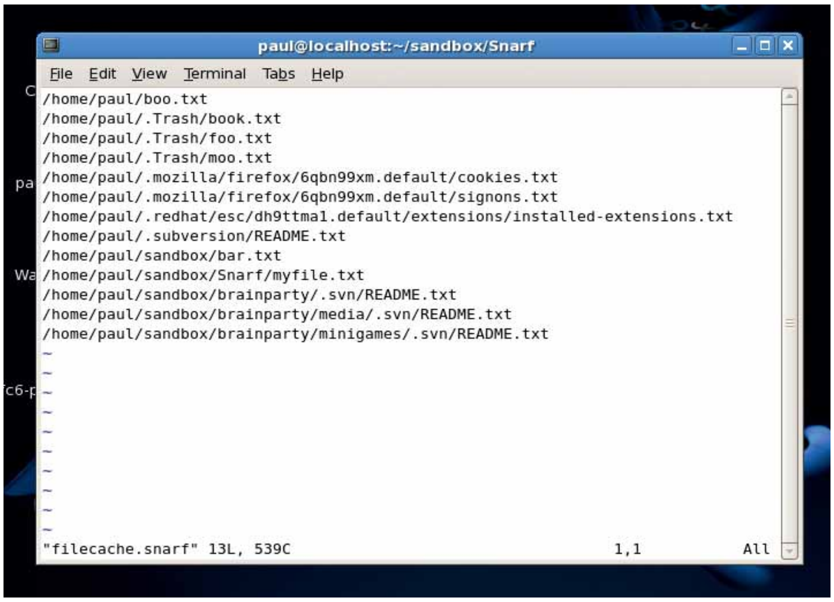
Используя одиночный знак = в условных выражениях, вы рискуете нарваться на проблемы. Например, рассмотрим код

```
if (Name = "Snarf") {
```

Этот код на самом деле означает «присвоить "**Snarf**" переменной Name, и если это произойдет удачно, то выполнить код внутри фигурных скобок». Такое присваивание, естественно, проходит всегда, и в качестве возвращаемого значения вернется **true** и будет выведено сообщение. Если вы часто пишете =, вместо ==, то можете решить проблему, просто поменяв местами параметры. То есть, следующие два выражения делают одно и тоже:

```
if (Name == "Snarf") {
// или...
if ("Snarf" == Name) {
```

Разница состоит в том, что если вы наберете = вместо == во втором варианте, то *MonoDevelop* откажется компилировать программу, потому что нельзя присвоить строке переменную – "**Snarf**" всегда останется "**Snarf**", и никак иначе.



› Так будет выглядеть наш кэш файлов: в каждой строке по одному имени txt-файла.

```
Console.Write("Громо... ");
Console.Write("Громо... ");
Console.Write("Громо... ");
Console.Write("Громокот! Хо!\n");
```

Строка **«Громо»** не раз повторяется, поэтому для ее многократного вывода на экран мы можем запихать ее вовнутрь так называемого «цикла for»:

```
for (int i = 1; i <= 3; ++i) {
  Console.WriteLine("Громо... ");
}
Console.WriteLine("Громокот! Хо!\n");
```

Согласен, в этом примере оба варианта кода имеют те же четыре строчки; ну, а если пришлось бы выполнить операцию 100 раз? Или 100000? Заметим, что ++ это сокращение C# для выражения **+=1** – оно просто прибавляет единицу к выражению. C# предусматривает несколько разных циклов, и for – один из них.

Идем дальше

Сейчас мы расширим нашу программу таким образом, что она будет считывать все файлы в каталоге, и если у файла расширение **txt** – распечатывать его содержимое. Тут нужны и цикл, и условное выражение – о меч Завета, помоги мне постичь непостижимое!

```
string[] files = Directory.GetFiles("/home/paul");
foreach(string file in files) {
  if (file.EndsWith(".txt")) {
    Console.Write(File.ReadAllText(file));
  }
}
```

Здесь показано аж 5 нововведений, поэтому позвольте мне разбить все по этапам:

- Если мы передадим Directory.GetFiles() каталог в качестве единственного параметра, то он вернет нам массив строк (string[], помните?), содержащий все имена файлов этого каталога.
- Элементы почти всех массивов можно перебрать с помощью цикла foreach: он извлекает каждый элемент массива, а мы присваиваем значение элемента переменной. В нашем примере, мы заставляем Mono присваивать каждое имя файла строке 'file'.
- У каждой строки есть метод EndsWith(), который возвращает **true**, если строка заканчивается подстрокой, которую мы передали ей в качестве параметра. Если метод возвращает **true**, то мы выполняем код внутри фигурных скобок (Console.Write(...)).
- Вместо того, чтобы присвоить возвращаемое значение File.ReadAllText() другой строке, мы сразу же передаем его в метод Console.Write. Так делать можно, и это помогает сделать код чуть короче.
- Отметим, что C# различает File (особый класс, позволяющий читать и записывать файлы) и file, строковую переменную, которую мы создали. Все переменные в C# чувствительны к регистру.

Замените содержимое метода Main() новым кодом, подставьте вместо **/home/paul** ваш собственный каталог с текстовыми **.txt**-файлами. Нажмите F5: вы увидите, что все работает, но это скорее код Громомальчика, чем Громомужа.

Вы когда-либо слышали фразу «быстро, качественно, дешево – выбери любые два»? Что ж, раз Linux связан с Open Source, «дешевизна» присутствует по определению. Но вот чудо: Mono позволяет еще и сделать «быстро» и «качественно»! Немного поколдуем, чтоб сделать наш код более быстрым и более функциональным.

Колдовство сидит в методе Directory.GetFiles(). Сейчас мы передаем ему один параметр, то есть каталог, где хотим искать файлы. Но в C# методы могут выполнять разные действия в зависимости от числа передаваемых параметров. Мы можем ускорить работу нашего кода, задав второй параметр метода GetFiles(), который позволит нам определить фильтр поиска для наших файлов. А именно, вместо того, чтобы использовать file.EndsWith(".txt"), используем второй параметр метода Directory.GetFiles, то есть *.txt. Тогда массив строк files будет содержать только файлы, заканчивающиеся на **.txt**. Можно заставить наш код делать даже больше, определив еще один параметр метода Directory.GetFiles, а именно SearchOption.AllDirectories. Параметр заставляет Mono искать не только в указанном каталоге, но и во всех его подкаталогах.

Поэтому новый супер-пупер метод Main() будет выглядеть следующим образом:

```
string[] files = Directory.GetFiles("/home/paul", "*.txt", SearchOption.
AllDirectories);
```

Скорая помощь

Если хотите красиво обработать несколько параметров командной строки, перестаньте использовать конструкцию args[0]. Вместо этого попробуйте использовать цикл foreach для извлечения каждого параметра командной строки и его последующей обработки.

Говорите в скобках!

Как мы видели в прошлый раз, C# широко использует фигурные скобки { и } как маркеры блоков – они отмечают начало и конец блоков кода. Хотя они необходимы для пространств имен, классов и методов, после условных выражений и циклов их использовать не обязательно. В данном случае скобки описывают, какой код исполнит C#, если условное выражение будет истинно или пока условие цикла истинно. При отсутствии скобок выполнится только одно выражение; если оно и вправду одно, зачем писать скобки? Вот, например:

```
if (Name == "Snarf") {
  Console.WriteLine("Snarf snarf!");
}
```

С тем же результатом можно использовать следующий код:

```
if (Name == "Snarf") Console.WriteLine("Snarf snarf!");
```

В этом коде метод WriteLine() выполнится только тогда, когда Name будет содержать строку **«Snarf»**. А все последующие выражения будут выполнены несмотря ни на что – даже если они расположены на той же строке, что вызов WriteLine(). Использование фигурных скобок рекомендуется при обучении: можно наглядно видеть, какой код относится к циклам и условным выражениям.

```
foreach (string file in files) {
  Console.Write(File.ReadAllText(file));
}
```

Эффектный финал

Места в статье уже не хватает – поэтому пора просить древних духов C# превратить этот загнивший код во что-то действительно работающее! Мы уже видели, как можно получить все имена файлов в данном каталоге (и его подкаталогах), и знаем, как читать и записывать файлы. Теперь нам необходимо, чтобы программа выполняла две вещи:

- Если параметры не указаны, то просканировать файловую систему и сохранить список в файле. То есть кэше файлов.
- Если параметр указан, то использовать его для поиска подходящих файлов, а затем вывести их содержимое на печать.

Наша программа будет делать нечто очень похожее на работу Linux-команды *updatedb*, используемую для генерации кэша поиска для *locate*. Но *updatedb* не выводит содержимого найденных файлов, значит, наша программа хоть чуть-чуть, да получше!

Вы уже знаете большую часть кода, которая необходима для выполнения этой работы, поэтому давайте сосредоточимся и начнем кодировать. Замените ваш метод Main() следующим кодом:

```
if (args.Length == 0) {
  string[] files = Directory.GetFiles("/home/paul", "*.txt", SearchOption.
AllDirectories);
  File.WriteAllLines("filecache.snarf", files);
} else {
  string[] cache = File.ReadAllLines("filecache.snarf");
  foreach(string file in cache) {
    if (file.Contains(args[0])) {
      Console.Write(File.ReadAllText(file));
    }
  }
}
```

Этот код содержит нечто совершенно новое: args.Length. В прошлый раз мы видели, что C# передает аргументы в метод Main() через массив строк 'args' (помните, что массив – это группа объектов одного типа). args.Length используется, чтобы узнать размер массива (т.е. сколько параметров было передано). Если он равен 0, то есть параметры не передавались, то нам необходимо создать кэш имен файлов.

Метод WriteAllLines() очень похож на метод WriteAllText(), за исключением того, что он принимает в качестве второго параметра не одну строку, а массив строк. Нам это очень полезно, так как Directory.GetFiles() возвращает массив строк, поэтому мы можем просто передать его в WriteAllLines(), нам не понадобится осуществлять построчную запись.

И опять мы встречаем нечто новое: else. Вы уже знакомы с if, условным оператором, выполняющим код при истинности условия. А что если условие ложно? Здесь приходит на помощь else. Например, следующий код выведет «Ты Громокот!»:

```
string Home = "Третья планета";
if (Home == "Средиземье") {
  Console.WriteLine("Ты хоббит!");
} else {
  Console.WriteLine("Ты Громокот!");
}
```

В нашей программе *Snarf* выражение else означает «если число передаваемых параметров не 0», то есть параметры были переданы. В этом случае вызывается метод ReadAllLines(), который считывает каждую строку текстового файла в строковый массив.

Наконец, мы переходим к главному блоку кода: мы просматриваем каждую строку в кэше файлов и проверяем ее на соответствие передаваемому параметру. Соответствие осуществляется с помощью специального метода Contains(), выполняемого над строками: передаем в качестве параметра подстроку, а метод возвращает true, если она содержится в строке. Переменная args[0], как вы узнали из прошлого раза, содержит первый параметр, переданный в командной строке. Если имя файла соответствует параметру, то мы считываем текст файла и выводим его на экран. Между прочим, кроме EndsWith() и Contains(), к строкам применимы методы Replace() (заменить одну подстроку другой), ToUpper() (преобразовать строку в верхний регистр) и Trim() (чтобы удалить пробелы, символы табуляции и символы новой строки, расположенные в начале и в конце строки).

Вот и все: наш проект закончен. *Snarf* роется в файловой системе, выводит на экран все файлы, соответствующие запросу, а Третья планета в очередной раз спасена от Мымм-Ра – и все благодаря Mono! На диске к журналу вы найдете полный код *Snarf*, а также дополнительные материалы. Обратите внимание на сообщения, которые выводятся при создании кэша файлов; numfilesfound – это переменная, которая увеличивается каждый раз при нахождении подходящего файла, чтоб мы могли вывести сообщение, если ни одного файла не обнаружится; а также, в случае успеха будет напечатаны имена всех файлов. LXF

Состояние .NET

На данный момент Microsoft выпустила три версии .NET: 1.0, 1.1 и 2.0. Mono – свободная реализация .NET – поддерживает почти всю реализацию 1.0 и 1.1, и в большинстве случае вы можете на это рассчитывать. В новые версии Mono добавляются многие новые возможности .NET 2.0, включая новые виджеты *Windows*.Forms (строительные блоки пользовательского интерфейса в Windows), а также новые методы, например, ReadAllText() и WriteAllText(), которые мы использовали на этом уроке.

Хотя .NET 2.0 была выпущена только в ноябре 2005, Microsoft уже готова выпустить .NET 3.0, капитальное изменение платформы. Не беспокойтесь: внутренности по большей части останутся прежними. Видите ли, .NET 3.0 был изначально известен как WinFX, и был спроектирован на основе .NET 2.0 с добавлением новых библиотек для поддержки нового пользовательского интерфейса, коммуникаций и тому подобных вещей. Фактически, .NET 3.0 – это .NET 2.0 плюс новый набор библиотек, и Mono не обязательно их реализовывать, если они не требуются.

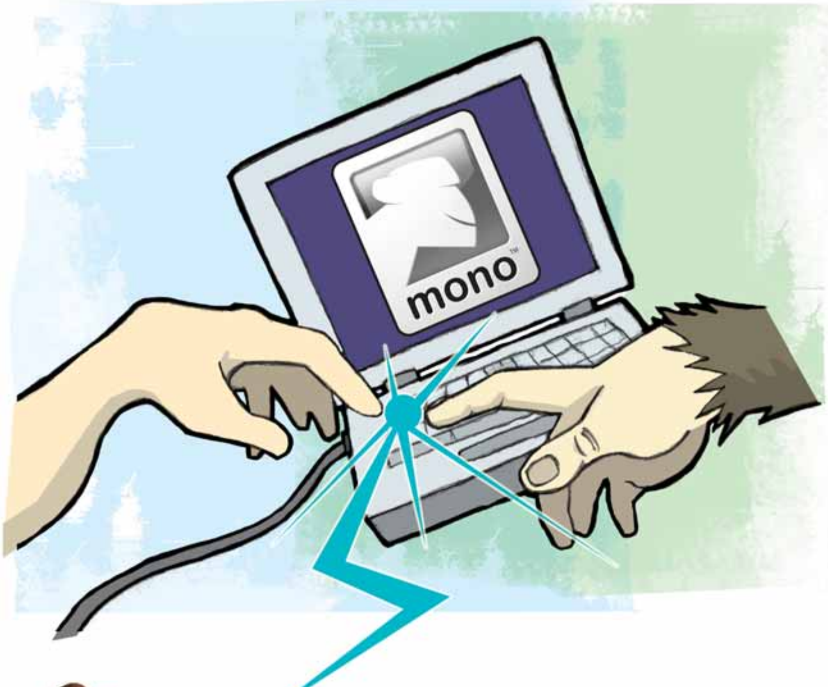
Скорая помощь

Наш код генерирует **filecache. snarf**, когда запускается без параметров, но вдруг пользователь сначала захочет запустить ее с параметром? В данной ситуации программа даст сбой, так как файла не было создано. Решение – использовать метод File. Exists (): если файл существует, то запускаем поиск. Если нет, то сначала создаем его и потом запускаем поиск.

Очень скоро Мы разберемся с XML и создадим программу чтения RSS-лент.

Mono: Читаем

На минутку оторвались от новостей на Slashdot, и уже боитесь, что отстали от жизни? **Пол Хадсон** говорит: прочтите обо всем с Mono – это проще, чем сосчитать до трех...





Наш эксперт

Пол Хадсон полагает, что Mono – лучшая вещь со времен мультфильма *Pinky* and the Brain, и сейчас поддерживает два проекта на основе Mono на SourceForge.

Интернет завален хламом – блогами людей, страдающих от своих детских страхов, бесчисленными «разборками» фанатов и миллионами часов видео про котят на YouTube. Возникает проблема: как пробраться через этот хлам к вещам действительно интересным? Поможет RSS: эта технология позволяет людям подписаться на интересующие их сайты и затем получать обновления по мере появления изменений. Например, если вы любитель новостей с сайта BBC News, но вам неохота что ни час лазать на него в поисках свежатинки, то RSS – для вас.

В прошлый раз мы разобрались, как работать с файлами, а теперь зайдем дальше: поработаем с XML. Формат XML очень похож на HTML: здесь тоже особым образом помечаются блоки текста. Но, в отличие от HTML, XML определяет не внешний вид текста, а его содержание, и это делает XML идеальным средством для пересылки данных. XML используется во множестве форматов, в том числе и RSS: Really Simple Syndication [очень простое приобретение информации]. RSS определяет способ хранения ленты новостей, обновляемой при каждом изменении на сайте; вы можете скомандовать компьютеру обновлять RSS-ленту каждые десять минут и читать заголовки новостей, не запуская web-браузер.

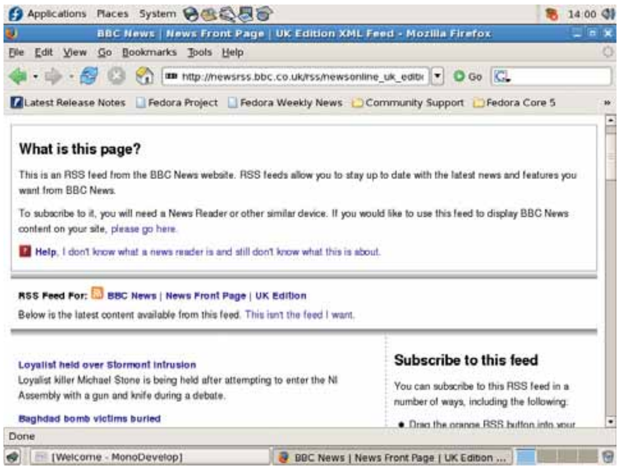
Mono поставляется с массой инструментов работы с XML, и нам не придется беспокоиться о том, как прочесть файл. Значит, можно сконцентрироваться на том, что мы хотим сделать, а именно:

0 Создать программу для скачивания и красивого вывода RSS-лент.

1 Заставить программу отслеживать содержимое ленты и отображать изменения, произошедшие с момента последнего просмотра.

2 Заставить программу сохранять выбранные пользователем ленты новостей.

Как и в прошлый раз, постараемся создать действительно полезную программу. Вперед!



› BBC.co.uk применяет таблицу стилей XSL к своей ленте RSS – для ее просмотра в неформатированном виде используйте пункт «Просмотреть исходный код» в контекстном меню браузера.

Охота на RSS

Во-первых, необходимо ясно понять, как выглядит RSS. Ниже представлена RSS-лента, и вы можете видеть, что у канала (channel, новостная лента) есть заголовок, описание и ссылка. Но это скорее метаинформация – если вас интересуют только новости, можете ее игнорировать. Вы также видите два элемента <item>, хотя их может быть сотня и более, в зависимости от ленты RSS. Эти элементы и есть пункты новостей, содержащие заголовок, описание и ссылки, на сей раз к конкретным новостям. Вот пример:

```
<?xml version="1.0" ?>
<rss version="2.0">
  <channel>
    <title>Мой превосходный сайт</title>
    <description>Здесь представлено много интересного – не забудьте подписаться!</description>
    <link>http://www.example.com</link>
    <item>
      <title>Mono рулит!</title>
      <description>Бесплатная реализация .NET покоряет мир</description>
      <link>http://www.example.com/news/mono</link>
      <guid>http://www.example.com/news/mono</guid>
    </item>
    <item>
      <title>Mono побеждает PHP</title>
```

» **Только что** Мы научились читать и записывать файлы с помощью Thundercats.

RSS-ленты




```
    <description>Согласованные функции – это круто</description>
    <link>http://www.example.com/news/monovsphp</link>
    <guid>http://www.example.com/news/monovsphp</guid>
   </item>
  </channel>
</rss>
```

Вы заметите, что у каждого пункта <item> есть элементы <link> и <guid>. 'GUID' (сокращение от Globally Unique Id, уникальный глобальный идентификатор) – любое значение, уникальное для данной новости во всей Сети. Этот пункт обязателен для лент RSS, так как именно по нему программы RSS определяют, видели они новость или нет. Будьте внимательны: GUID обязан быть уникальным не только на своем сайте, но и на других. Простейший (и наиболее распространенный) способ выбора GUID – сделать его ссылкой на новость: уникальность гарантируется.

Рассмотрим реальный пример. Начните новый консольный проект в *MonoDevelop* и назовите его по своему вкусу. Наверху, где находятся строки 'using', добавьте следующую:

```
using System.XML;
```

Далее, на панели Solution (слева в окне *MonoDevelop*), правой кнопкой щелкните на References и в появившемся меню выберите Edit References. Убедитесь, что там выбрано System.XML, и нажмите OK. Теперь приведите метод Main() к следующему виду:

```
XmlDocument doc = new XmlDocument();
doc.Load("http://tinyurl.com/8mwkm");
Console.Write(doc.InnerXml);
```

Ссылка TinyURL вставлена для экономии места – если угодно, укажите полный URL: **http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss.xml**.

Этот код использует класс XmlDocument для чтения URL и вывода его на экран. Пока мы ничего особенно не делаем: просто выводим документ на консоль. Нажмите F5, чтобы скомпилировать и запустить программу. Вы увидите в окне Application Output в *MonoDevelop* большой кусок текста. Это наша RSS-лента – не беда, что она выглядит довольно зверообразно: мы ее приручим!

Просто заголовки

Для дрессировки RSS есть отличные методы .NET: SelectSingleNode() и SelectNodes(). Они позволяют осуществлять поиск нужных данных в XML-документе и извлекать либо самый первый XML-узел (т.е. XML-элемент <item>, прочитанный нашей программой), либо все подходящие узлы.

Итак, мы хотим, чтобы версия номер два нашей программы считывала все новости и для каждой выводила ее заголовок и описание. Вот мой рецепт для *Hudson RSS Reader v2*:

0 Пропустить RSS через XmlDocument.load().

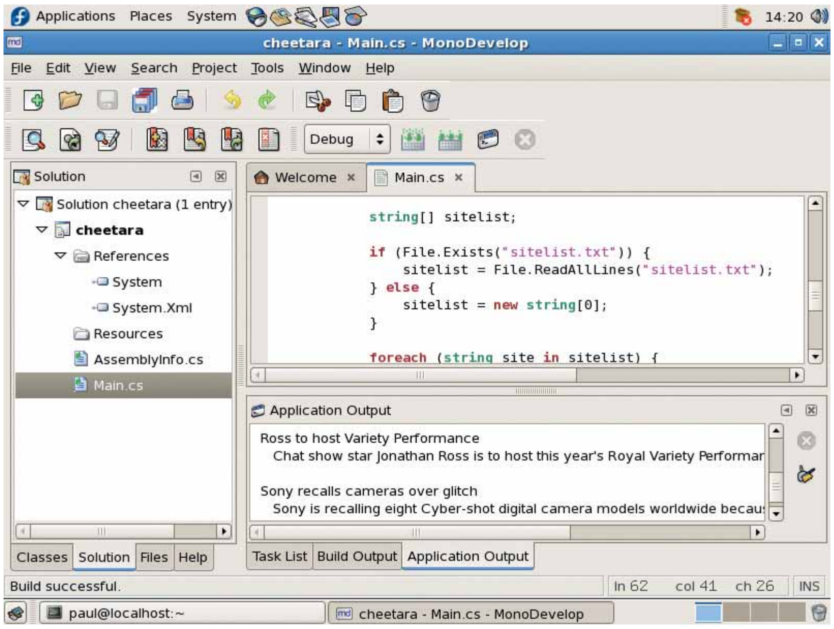
1 Отделить интересующие нас элементы <item>.

2 Просеять элементы с предыдущего шага и при необходимости откинуть их данные на Console.Write().

3 Подсолить и подать на стол.

Или – по-простому, в синтаксисе C#...

```
XmlDocument doc = new XmlDocument();
doc.Load("http://tinyurl.com/8mwkm");
XmlNodeList items = doc.SelectNodes("//item");
foreach (XmlNode item in items) {
  Console.WriteLine(item.SelectSingleNode("title").InnerText);
  Console.WriteLine(" " + item.SelectSingleNode("description").InnerText);
  Console.WriteLine("");
}
```



› Среда *MonoDevelop* превосходна для кодирования, но нам придется использовать CLI, чтобы вставить собственные параметры. Обратите внимание на ссылку System.XML на левой панели!

Параметр, передаваемый в SelectNodes – //item – известен как Xpath. Это особый способ поиска элементов внутри XML, и в нашем случае он требует 'брать из XML-документа все элементы <item>'. Двойной слэш // именно и означает 'взять любое вхождение'. Взгляните на следующий XML-документ:

```
<stuff>
 <clothing>
  <item>Брюки</item>
  <item>Носки</item>
 </clothing>
 <news>
  <item>Релиз Wii состоялся</item>
  <item>Xbox 360 - отстой!</item>
 </news>
</stuff>
```

Извлекая из него элементы <item> при помощи Xpath //item, вы довольны не останетесь: ваш запрос свалит в кучу и новости, и штаны с носками! Вместо использования // 'найти любого', уточним запрос: скажем, что нам нужны элементы <item>, являющиеся частью элементов <news>. В XPath это выглядит как /news/item.

Впрочем, RSS-лента использует элементы <item> только для обозначения новостей, поэтому использовать //item вполне безопасно. Этот поиск возвращает переменную, известную как XmlNodeList. Если XML-узел (XML node) содержит один XML-элемент, то в XmlNodeList, видимо, находится несколько XML-элементов, верно? Верно. Я просто хотел убедиться, что вы не запутались при обсуждении Xpath!

Раз уж список новостей получен, остается только их распечатать. В прошлый раз я познакомил вас с циклом foreach, теперь он снова вернулся – и будет работать с XmlNodes, а не просто со строками. Этот цикл перебирает каждую новость, которую возвращает »

Скорая помощь

Для создания файла есть метод File.Create(), но мы обойдемся без него – метод File.AppendAllText() автоматически создаст файл, если его не существует.

» метод SelectNodes(), и присваивает ее переменной item, которую мы можем прочесть. Каждый пункт <item> в нашем XML-документе содержит несколько интересных подпунктов: заголовок новости, описание, ссылку и т.д. Чтобы извлечь любой из них, необходимо применить к нашему пункту метод SelectSingleNode, который возвращает XmlNode. Например, чтобы получить заголовок новости, необходимо применить item.SelectSingleNode("title"). Однако метод вернет просто XML-узел, являющийся .NET-представлением XML-элемента <item>, а не содержание XML-узла. Тут пригодится InnerText: он возвращает текстовое содержимое объекта XmlNode.

Держа все это в голове, посмотрим снова на эту строчку:

```
Console.WriteLine(item.SelectSingleNode("title").InnerText);
```

Она работает так: берет текущий узел, затем его заголовок и текстовое содержимое, и выводит на консоль.

Когда все это выведено, вызывается Console.WriteLine() для вывода пустой строки между новостями.

Вот и все – скомпилируйте и запустите вашу программу с помощью F5 и наслаждайтесь: ваши кулинарные навыки превратили сырые ингредиенты RSS-ленты в вывод на вашем экране!

Что нового, малыш?

У нашей программы есть проблема: RSS-ленты бывают очень большими, а людям подавай только свежие новости, появившиеся после последнего опроса ленты. Это действительно проблема: как уследить, что некоторые новости уже прочитаны, и показать только те, которые вы еще не видели? Что ж, возвратитесь назад на 1000 слов, к уникальным глобальным идентификаторам. Вот что я говорил: «Этот пункт обязателен для лент RSS, так как именно по нему программы RSS определяют, видели они новость или нет.» Каждой RSS-новости необходим GUID, однозначно определяющий ее в сети, и его можно использовать для решения проблемы.

Вот алгоритм:

0 Получить ленту RSS.

1 Сохранить все GUID в файле, по одному на строку.

2 В следующий раз, когда будет загружена RSS-лента, будут показаны только новые новости, отсутствующие в кэше GUID.

Всего три шага, но их кодирование немного сложнее. Вот как будет выглядеть метод Main() – я добавил комментарии, поясняющие, как все работает:

```
XmlDocument doc = new XmlDocument();
doc.Load("http://tinyurl.com/8mwkm");
// Этот массив строк будет хранить содержимое кэш-файла
string[] guidcache;
if (File.Exists("guidcache.txt")) {
  // У нас есть кэш-файл – так прочтем его!
  guidcache = File.ReadAllLines("guidcache.txt");
} else {
  // У нас нет кэш-файла – создадим новый массив строк из 0
элементов (то есть пустой)
  guidcache = new string[0];
}
// Получить все новости…
XmlNodeList items = doc.SelectNodes("//item");
foreach (XmlNode item in items) {
  // Положим, что мы собираемся показать эту новость по умолчанию
  bool showthisitem = true;
  // Теперь переберем каждый GUID в кэше…
  foreach (string guid in guidcache) {
    // … и сравним его с GUID этой новости.
    if (guid == item.SelectSingleNode("guid").InnerText) {
      // Если у нас есть совпадение – то не показываем эту новость!
      showthisitem = false;
      // Следующее выражение говорит C# выйти из цикла – мы уже
нашли GUID,
      // поэтому не надо проверять остальной кэш GUID.
      break;
    }
  }
  if (showthisitem) {
    // Сюда попадаем только если GUID не находится в нашем кэше –
выводим его на экран!
    Console.WriteLine(item.SelectSingleNode("title").InnerText);
    Console.WriteLine(" " + item.SelectSingleNode("description").
InnerText);
    Console.WriteLine("");
    // … теперь добавим GUID в конец кэш-файла.
    File.AppendAllText("guidcache.txt", item.SelectSingleNode("guid").
InnerText + "\n");
  }
}
```

Скорая помощь

Очень важно проверять, сколько аргументов передается в массив args, прежде чем работать с ним, потому что Mono вылетит, если вы попытаетесь прочитать элемент, которого не существует. Будьте внимательны!

Это простейший вариант кода, но если вы ищете нечто побыстрее в работе, предлагаю вам вставить следующую строку сразу после выражения bool showthisitem:

```
string thisguid = item.SelectSingleNode("guid").InnerText;
```

Вместо того, чтобы вызывать SelectSingleNode() для каждого GUID в кэше и для каждой новости, этот код кэширует GUID в строке символов, и вы можете использовать ее вместо вызовов SelectSingleNode().

Подпишись сегодня!

Давайте разгоним нашу программу: сейчас в нашем исходном коде навеки прописан URL BBC. А вдруг люди захотят увидеть другие новостные источники? Или читать несколько новостных сайтов и одновременно обновлять их? Для этого требуется кодирование посложнее, но тут-то наша программа и станет по-настоящему полезной.

Вот какой я ее вижу:

0 Когда программа запускается с параметром sub, за которым следует URL, она должна подписаться на эту ленту.

1 Когда программа запускается с параметром unsub, за которым следует URL, она должна отписаться от данной ленты.

2 Когда она запускается без параметров, то должна обновить все RSS-ленты и показать все новые записи.

3 Когда программа запускается с параметром reset, она должна очистить список GUID и обновить ленты, показав все записи во всех лентах.

В общем, не очень далеко от уже готового кода, но есть одна особенность: пунктам 2 и 3 необходимо выводить на экран RSS-ленты. Самый тупой способ это сделать – взять кусок кода с печатью ленты, скопировать его и вставить второй раз, но куда лучше создать отдельный метод: его можно вызывать откуда угодно, и код будет сосредоточен в одном месте.

Но сначала надо написать код для подписки на новостные ленты и отказа от нее. Здесь вы можете изучить новую классную штуку – блок switch/case. Вам уже попадалось условное выражение (вспомните: if/else), однако его трудновато использовать при проверке множества условий. Блок switch/case позволяет проверять значение переменной, не засоряя код. Например, простейший код для проверки того, что должна делать программа, выглядит так:

```
switch (args.Length) {
  case 0:
    // обновить ленты!
    break;
  case 1:
    // осуществить сброс лент!
    break;
  case 2:
    // подписаться или отписаться на новостные ленты!
    break;
}
```

Это код проверяет значение переменной args.Length на равенство **0**, **1** или **2**. Значение args.Length автоматически устанавливается равным числу параметров, передаваемых программе из командной строки, поэтому фактически оно определяет «Как эта программа была

запущена?» Если параметры отсутствуют, значит, достаточно обновить ленты новостей. Если имеется один параметр, можно сразу выполнить сброс лент новостей, даже не проверяя, что это за параметр: единственное действие, описываемое одиночным параметром – перегрузка новостных лент [в «настоящей» программе не помешало бы сделать проверку, что этот единственный параметр – 'reset'; никто не запрещает пользователю запустить программу с единственным параметром 'sub' или даже 'asdfghj'! В последних двух случаях его следует известить об ошибке – например, распечатать список допустимых опций командной строки, – прим. ред.]. Наконец, если передается два параметра, то необходимо проверить, sub это или unsub, а затем предпринять соответствующее действие.

Принимаем решение

Когда switch натыкается на совпадение, то выполняет код, начиная с найденного места, пока не упрется в другую строку case или инструкцию break. Строка 'break' просто сообщает: «С совпадением покончено, переходи в конец блока switch/case.» В предшественнике C# – C++, если вы не ставили инструкцию break, программа продолжала выполнение следующего выражения case, вне зависимости от того, произошло совпадение или нет. В C# этого не происходит, но break все равно является обязательным.

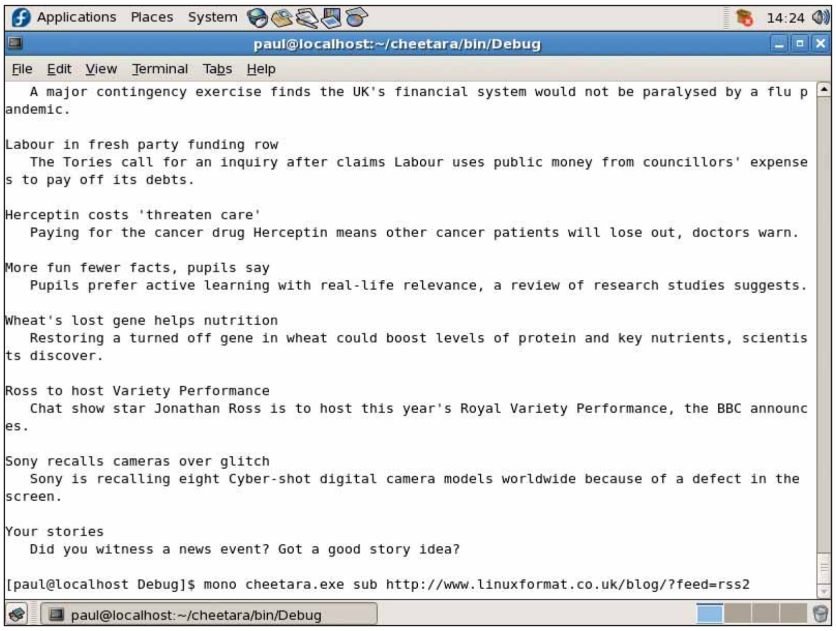
Сначала разберемся с подпиской и отказом от лент новостей. Для этого требуется проверить, что аргумент – либо sub, либо unsub, затем добавить ленту в список подписки. Вот как это происходит:

```
case 2:
 // Подпишись или откажись!
 if (args[1] == “”) return;
 if (args[0] == “sub”) {
  // Добавить сайт в существующий список
  File.AppendAllText(“sitelist.txt”, args[1] + “\n”);
 } else {
  if (File.Exists(“sitelist.txt”)) {
   // Удалить сайт из списка
   string[] sitelist = File.ReadAllLines(“sitelist.txt”);
   File.Delete(“sitelist.txt”);
   foreach (string site in sitelist) {
    if (site == args[1]) {
    // Да! именно этот сайт надо удалить; игнорируем его
    } else {
     File.AppendAllText(“sitelist.txt”, site + “\n”);
    }
   }
  }
 }
 break;
```

Подписка выглядит довольно просто, а вот с отказом все обстоит немного сложнее. В приведенном выше коде сначала происходит чтение файла с сайтами, затем он удаляется. После этого мы перебираем все сайты, на которые подписался пользователь, и сохраняем их по одному на строку в файл. А когда встретим сайт, от которого хотим отказаться, мы его пропускаем. Понятно?

Остальные два случая проще и выглядят так:

```
switch (args.Length) {
 case 0:
  // Обновить!
  ReadFeeds();
  break;
 case 1:
  // Сброс!
  File.Delete(“guidcache.txt”);
  ReadFeeds();
  break;
```



› **Готовый продукт: Программа чтения RSS-лент, помнящая, что вы читали.**

Метод ReadFeeds() – то самое повторное использование кода, о котором я говорил: можно было вставить весь код для чтения новостей прямо в case-выражение, но гораздо лучше создать собственный метод, ReadFeeds(). Итак, если ваша программа выполняется без параметров, сразу же вызывается метод ReadFeeds(); а если с параметром, мы очищаем кэш GUID, а уж потом вызываем ReadFeeds().

Метод ReadFeeds() в принципе такой же, что и старый код для чтения RSS, но я боюсь, что его придется модифицировать, если вы захотите читать ленты с нескольких сайтов. Для этого необходимо в списке подписки перебрать в цикле все сайты, а в каждом сайте – все его новости. Вот главная часть:

```
string[] sitelist;
if (File.Exists(“sitelist.txt”)) {
 sitelist = File.ReadAllLines(“sitelist.txt”);
} else {
 sitelist = new string[0];
}
foreach (string site in sitelist) {
 XmlDocument doc = new XmlDocument();
 doc.Load(site);
```

Здесь нет ничего нового, но этот код идеально завершает нашу программу. Нажмите F8, чтобы скомпилировать ее, затем откройте терминал и перейдите туда, где расположен проект *MonoDevelop*. Далее зайдите в каталог **bin/Debug**, там находится исполняемый файл. Запустите его – и, я думаю, вы согласитесь, что программа и вправду полезна!

Мы прошли немалый путь: вы больше не боитесь XML, вы поняли, что создавать свои методы – это хороший стиль, и изучили блок switch/case, благодаря которому условные выражения выглядят опрятно. А главное, вы создали свое второе полезное рабочее приложение на C# с помощью Mono – молодцы! LXF

Скорая помощь

Мы используем параметр reset, чтобы очистить GUID'ы для всех наших RSS-лент – тогда наша программа скачает все новости из новостных лент. Если вы хотите разрешить людям предоставлять параметр URL для reset, чтобы сбросить только конкретную ленту, то самый простой способ это сделать – хранить GUID в отдельных файлах: вместо **guidcache.txt** у вас может быть **guid-news.bbc.co.uk.txt**. Чтобы очистить кэш GUID для одного сайта, просто удалите соответствующий ему файл.

» **Через месяц** Здравствуй, *Beagle*! Мы подцепим некоторые занятные библиотеки Mono.

Настоящая безопасность Как воспользоваться умными утилитами Linux и защитить вашу машину.

LINUX FORMAT на DVD
» Aide 0.12
» Tripwire 2.4.0.1

Безопасность:

ЧАСТЬ 5: Д-р Крис Браун заканчивает серию уроком обнаружения вторжений с помощью Aide и Tripwire – для администраторов это вроде дактилоскопии и, простите за милитаризм, сигнальной мины.





Наш эксперт

Д-р Крис Браун независимый инструктор по Linux, имеет степень доктора наук по физике элементарных частиц, сертифицированный специалист Novell и Red Hat. Недавно написал книгу о SUSE для издательства O'Reilly.

Свойства файла в Aide

Свойство	Значение
p	Права доступа
i	Номер inode
n	Количество ссылок
u	Пользователь (вледелец)
g	Группа
s	Размер
b	Количество блоков
m	Время последнего изменения
a	Время последнего доступа
c	Время последнего изменения статуса
s	Проверка на увеличение размера
md5	Контрольная сумма md5
sha1	Контрольная сумма sha1
R	p+i+n+u+g+s+m+c+md5
L	p+i+n+u+g
E	Пустая группа

› **Рис. 1 Вы можете указать *Aide* те Свойства файлов, изменения которых вы считаете подозрительными.**

Как узнать, что вас «хакнули»? Это может быть неочевидно. Непрошеные гости не всегда портят сайты или вводят «здесь был Вася» в **/etc/motd**. Они могут втихую использовать ваш компьютер как, например, ретранслятор почты для рассылки спама.

На четырех предыдущих уроках мы рассмотрели активные способы повышения безопасности Linux-систем – другими словами, предотвращения взлома. На этом, финальном, уроке займемся парой инструментов, *Tripwire* и *Aide*, созданных для отслеживания несанкционированных изменений файловой системы. Эти инструменты по своей природе скорее реактивны – они не предотвращают атаку. Зато они смогут сообщить вам, что ваша безопасность под угрозой – и, уж поверьте мне, лучше знать об этом, чем не знать. Я научу вас настраивать их так, чтобы регулярные проверки не отвлекали ваше внимание на нормальные события.

Основа работы *Tripwire* и *Aide* одна: делается «снимок» состояния файловой системы (по крайней мере, значимых ее частей) в тот момент, когда есть уверенность в ее нетронутом состоянии. Затем с регулярными интервалами (желательно ежедневно, через расписание *Cron*) программа запускается снова, и состояние файловой системы сравнивается с исходным снимком. Составляется отчет с описанием изменений. Отчет можно сохранить на диск, а можно отослать по почте системному администратору. Учтите, эти отчеты действительно придется просматривать, если вы хотите, чтобы от инструментов обнаружения вторжений был прок. Вдумчивое использование *grep* может устранить некоторые излишества, но в конечном итоге отчет должен читать конкретный человек.

Сначала Aide

Начнем с *Aide*. Аббревиатура означает Automatic Intrusion Detection Environment (среда автоматического обнаружения вторжений), написали программу Рами Лехти [Rami Lehti] и Пабло Виролайнен [Pablo Virolainen]. Лехти учился в технологическом университете Тампере (Финляндия), и копию учебника по *Aide* до сих пор можно найти на университетском сайте по адресу: www.cs.tut.fi/~rammer/Aide/manual.html. Позднее проект был подхвачен Ричардом ван ден Бергом [Richard van den Berg] и Майком Маркли [Mike Markley] и приютился на http://sourceforge.net/projects/Aide. При желании оттуда можно загрузить исходный код. Однако для подготовки этого урока я не стал возиться с исходными текстами, а взял готовый пакет из состава SUSE Linux Enterprise Desktop 10 (можете воспользоваться любой другой свежей версией SUSE).

Ни одна уважающая себя программа не обходится без файла настроек. *Aide* – не исключение, но стандартная конфигурация совсем неплоха, так что перейдем сразу к делу (а с файлом разберемся позже). Для начала необходимо создать исходный снимок. Команда проста: *aide --init*. Делать это нужно тогда, когда вы уверены в целостности системы: например, сразу же после инсталляции с надежного носителя и начальной настройки, но перед подключением машины к сети (или, что случается чаще, перед продолжительным выходом в Интернет для

» **Месяц назад** Мы построили межсетевой экран Linux с использованием iptables.

Ищем лазутчиков

закачки обновлений, предложенных дистрибутивом). *Aide* исследует файловую систему полностью, поэтому процесс занимает несколько минут.

Результат работы программы можно будет найти в **/var/lib/Aide/Aide.db.new**. Это обычный текстовый файл, а не двоичная копия всей файловой системы (а также не снимок логического тома, пригодный для отката к прежнему состоянию). Здесь содержится по строке на каждый файл, в которой записаны сведения об имени файла, правах доступа, принадлежности к пользователю и группе, размере, времени последнего изменения, времени последнего изменения статуса, номере inode, числе связей, а также один или больше дайджестов (криптографических хэшей) его содержания. Дайджесты дают возможность *Aide* отслеживать изменения в содержимом файла.

Теперь снимок нужно переименовать в **aide.db**, поскольку именно этот файл *Aide* будет разыскивать впоследствии:

```
# cd /var/lib/aide
# mv aide.db.new aide.db
```

Представьте, что удаленный взломщик поместил программу под названием *cal* в директорию **/bin**. А в Linux есть стандартная и совершенно безвредная программа с таким именем, так что мы и не заметим кукушонка, если не будем сверхбдительны и не запомним, что настоящая *cal* живет в **/usr/bin**. Смоделируем подозрительный процесс следующим образом:

```
# cp /bin/cp /bin/cal
```

Теперь запустим *Aide* в проверочном режиме (в реальности, повторимся, это следует делать через *Cron*):

```
# aide --check -V2
Aide found differences between database and filesystem!!
Start timestamp: 2006-10-03 05:56:39
Summary:
  Total number of files:      149004
  Added files:                2
  Removed files:              1
  Changed files:              3
---------------------------
Added files:
---------------------------
added:/var/lib/aide/aide.db
added:/bin/cal
---------------------------
Removed files:
---------------------------
removed:/var/lib/aide/aide.db.new
---------------------------
Changed files:
---------------------------
changed:/etc/cups/certs
changed:/etc/cups/certs/0
changed:/bin
```

Нетрудно видеть, что *Aide* прилежно отметила добавление файла **/bin/cal**. Замечено также переименование **aide.db.new** в **aide.db**. Есть и третье сообщение, об изменении файла **/etc/cups/certs/0**. Что это? Честно говоря, сам не знаю. Это идет от системы печати CUPS, а обновляется каждые пять минут при любой погоде, но я не знаю, зачем. Мы узнали важную вещь: *Aide* присущи сигналы ложной тревоги (об изменениях, расцениваемых как подозрительные, но таковыми не являющимися).

Уточним поиск

В процессе повседневной работы файловая система меняется – как планово, так и неожиданно. Например, в домашней директории можно ожидать любых изменений; но системные утилиты вроде *ps* и *ls* не должны меняться никогда [разве что при крупном обновлении системы, – прим. ред.]. Вот и пришла пора заняться файлом настройки *Aide* (**/etc/Aide.conf**). Его основная задача – указать директории и файлы, за которыми нужно следить, а также свойства файлов, нуждающиеся в контроле. Как показано на **Рис. 1**, каждому параметру соответствует особый символ. Обратите внимание на параметры R и L: они определяются комбинацией базовых свойств. Можно задать и собственные комбинации в **aide.conf**, следующим образом:

```
Binlib = p+i+n+u+g+s+b+m+c+md5+sha1
Logs   = p+i+n+u+g+S
```



```
# Simple aide.conf file

# Configuration parameters
database=file:/var/lib/aide/aide.db
database_out=file:/var/lib/aide/aide.db.new
verbose=1
report_url=stdout
warn_dead_symlinks=yes

# Custom rules
#
Binlib        = p+i+n+u+g+s+b+m+c+md5+sha1
ConfFiles     = p+i+n+u+g+s+b+m+c+md5+sha1
Logs          = p+i+n+u+g+S
Devices       = p+i+n+u+g+s+b+c+md5+sha1
Databases     = p+n+u+g
StaticDir     = p+i+n+u+g
ManPages      = p+i+n+u+g+s+b+m+c+md5+sha1

# Kernel, system map, etc.
/boot                              Binlib

# watch config files, but exclude, what changes at boot time, ...
!/etc/mtab
!/etc/lvm*
/etc                               ConfFiles

# Binaries
/bin                               Binlib
/sbin                              Binlib

# Libraries
/lib                               Binlib

# Complete /usr and /opt
/usr                               Binlib
/opt                               Binlib

# Log files
/var/log$                          StaticDir
/var/log                           Logs

# Devices
!/dev/pts
/dev                               Devices

# Other miscellaneous files
/var/run$                          StaticDir
!/var/run
/var/lib                           Databases

# Test only the directory when dealing with /proc
/proc$                             StaticDir
!/proc
```

› **Рис. 2 Сердце вашей следящей системы: простой, но информативный файл aide.conf.**

Набор параметров Binlib (или называйте как хотите) объединяет в себе все свойства файла, включая его дайджест md5 и sha1, а также дайджесты. Взглянув на **Рис. 1**, мы заметим, что упущено лишь одно свойство, «a», время последнего доступа. Такой набор параметров подходит для системных утилит и библиотек, которые не должны изменяться никогда. Набор параметров Logs менее строг, он подойдет для файлов, которым позволено расти в размерах, например, файлов журналов.

Между прочим, назначение свойства «a» (времени последнего доступа) для меня загадка: я сомневаюсь, что оно когда-либо вам пригодится. Мне кажется, если к файлу нельзя даже притронуться, то проще зашифровать его или упрятать в безопасное место.

Главная часть файла конфигурации – правила, определяющие, какие части файловой системы и по каким критериям должна проверять *Aide*. Правила в основном совершенно несложные. Например, правило /lib Binlib означает, что *Aide* будет проверять всю файловую систему внутри **/lib** с помощью набора параметров Binlib, заданного ранее, и доложит обо всех приключившихся там изменениях. Левая часть правила (/lib в нашем случае) трактуется как регулярное выражение, которому должно удовлетворять имя файла (совпадение неявно привязано к началу имени). Таким образом, левая часть /lib будет соответствовать и **/lib/libc.so.6**, и **/lib/security/pam_unix**, но не будет соответствовать **/var/lib/locatedb**. Можно ограничить поиск, приписав »

Скорая помощь

Подход истинного параноика – запускать *Aide* с надежного CD, а файл **.conf** и базу данных моментальных снимков хранить на носителях в режиме «только для чтения».

Польза от слежки

Поиск нежелательных изменений в файловой системе может помочь не только обнаружить вторжения. Например, с его помощью в корпоративной сети можно поймать за руку пользователей, устанавливающих неразрешенное ПО или обновляющих свой компьютер. (За недолгую работу в некоем подобии корпоративной сети я постоянно воевал с IT-персоналом по этой причине.)

Основные компоненты Tripwire

Компонент	Описание
/sbin/tripwire	Основная программа *Tripwire*; создает исходную базу данных, впоследствии следит за ее изменениями.
/sbin/twadmin	Административный инструмент *Tripwire*.
/sbin/twprint	Инструмент просмотра защищенных цифровыми подписями файлов *Tripwire*.
/etc/tripwire/twcfg.txt	Текстовая версия конфигурационного файла *Tripwire*, в которой указывается местонахождение прочих операционных файлов.
/etc/tripwire/tw.cfg	Зашифрованная версия **twcfg.txt**
/etc/tripwire/twpol.xt	Файл политик *Tripwire*, в котором указываются файлы и директории для мониторинга, а также критерии проверки. Основная работа по настройке *Tripwire* сосредоточена здесь.
/etc/tripwire/tw.pol	Зашифрованная версия **twpol.txt**
/var/lib/tripwire/hostname.twd	База данных, содержащая снимок файловой системы (здесь, hostname нужно заменить реальным).
/Var/lib/tripwire/report	*Tripwire* сохраняет результаты проверок в этой директории, в файлах с именами вида **hostname - YYYYMMDD-HHMMSS. twr**.

› (Рис. 3) У *Tripwire* масса исполняемых и настроечных файлов, баз данных и отчетов. Это – основные..

» к концу имени знак $: например, левая часть выражения /tmp$ будет соответствовать **/tmp** и ничему иному.

Вывести названия из-под проверки можно, добавив префикс !. Например, правило !/.*/BACKUP исключит все директории **BACKUP**, независимо от расположения в файловой системе. Особо отмечу, что .* в этом примере – регулярное выражение, а не маска имени файла.

Пишите собственные правила

На **Рис. 2** приведен пример **Aide.conf**. В нем содержатся кое-какие глобальные настройки, усовершенствованные правила (наборы параметров) и несколько правил, определяющих те свойства файлов, которые нам хотелось бы видеть неизменными в каждой директории.

Пример конфигурационного файла *Aide* из SUSE Linux содержит «разумные» правила, и, конечно, лучше применять *Aide* со стандартным набором правил, чем не применять вообще. Но еще лучше поразмыслить над особенностями своей машины и выработать собственную политику безопасности и собственные наборы параметров. Процитирую разработчиков: «Неплохая идея – игнорировать постоянно меняющиеся директории, если вы не хотите получать слишком громоздкие отчеты. Лучше исключить временные директории, почтовые очереди, директории с файлами журналов, файловую систему **/proc** – все, что меняется регулярно. Проверять следует все системные двоичные файлы, библиотеки, include-файлы и исходные файлы системы».

Будучи не в меру мнительным (то есть запустив слишком частые или слишком строгие проверки), вы получите слишком много ложных сигналов, а в длиннющем отчете потонут указания на реальные проблемы, или, хуже того, отчет раздуется настолько, что вы вообще поленитесь его читать.

Aide может стать сердцевиной системы обнаружения вторжений, но это еще не комплексное решение. Неплохо будет добавить строчку в ваш файл *Cron* для ежедневного запуска *Aide*, а может быть, вы напишете сценарий оболочки для разумного распоряжения результатами (отправки кому-нибудь по почте?). Программа не лишена слабых мест, уязвимых для атак: ведь эрудированный злоумышленник может предполагать ее существование. Во-первых, сам исполняемый файл *Aide* может быть модифицирован с целью не допустить сообщения об инсталляции подозрительного руткита. Интервент может исказить файл конфигурации, исключив из него правила для «своих» файлов. Наконец, сметливый хакер может внедрить свое ПО, запустить *aide --init* вторично, и – концы в воду.

Беремся за Trip

Теперь обратим внимание на *Tripwire* [*Tripwire* – растяжка, – прим. пер.]. Если *Aide* твердо придерживается принципа открытости (и распространяется под GPL), *Tripwire* теперь принадлежит Tripwire Inc., где она развилась от простой программы проверки целостности файловой системы вроде *Aide* до (извините за рекламный пафос: я цитирую web-сайт) «всестороннего решения для аудита изменений, безопасности и соответствия для любых серверов Linux/Unix/Windows, баз данных, сетевых устройств, настольных машин и служб каталогов». Ого! Все это правда – только вот коммерческие версии *Tripwire* теперь далеки от центра внимания сообщества.

Но свободная версия *Tripwire* пока существует, и доступна по адресу: http://sourceforge.net/projects/Tripwire. Для целей данного урока я загрузил двоичную версию (файл **Tripwire-2.4.0.1-x86-bin.tar.bz2**) и установил ее в своей системе Fedora Core 5. Не обошлось без трудностей. При полном отсутствии хоть какого-то подобия инсталляционного скрипта, мне пришлось скопировать двоичные файлы, man-страницы и файлы конфигурации в нужные места вручную. В сети можно встретить *Tripwire* в виде различных RPM; инсталляция одного из них намного упростит вашу задачу.

Концепция *Tripwire* во многом схожа с *Aide*, но административно она более сложна. На **Рис. 3** показаны основные компоненты *Tripwire*. Усложнение частично происходит из-за того, что *Tripwire* шифрует свои файлы и снабжает цифровыми подписями, что устраняет нужду в защищенных носителях, предложенных мной для *Aide*. Поэтому существуют, например, текстовая и шифрованная версии конфигурационного файла, и такие же версии файла политики (в файле политики перечисляются файлы и директории для мониторинга, а также параметры проверки).

После инсталляции *Tripwire* необходимо создать общий (site-wide) и локальный (для каждого хоста) ключи шифрования. Для этого используйте *twadmin*, например вот так:

```
# twadmin -m G -S /etc/Tripwire/site.key
# twadmin -m G -L /etc/Tripwire/hostname-local.key
```

В каждом случае вам будет предложено ввести и подтвердить пароль, которым будет впоследствии открываться доступ. Общий ключ используется для защиты файлов, используемых на нескольких системах. Среди них – файлы конфигурации и файлы политик, мы до них скоро доберемся. Локальный ключ защищает файлы локальной машины, например, базу данных *Tripwire*. Локальный ключ можно использовать и для подписи проверочных отчетов.

У *Tripwire* два файла конфигурации. Первый, **/etc/Tripwire/twcfg.txt**, указывает расположение остальных операционных файлов *Tripwire*. Вот пример:

```
POLFILE     = /etc/Tripwire/tw.pol
DBFILE      = /var/lib/Tripwire/$(HOSTNAME).twd
REPORTFILE  = /var/lib/Tripwire/report/$(HOSTNAME)-
$(DATE).twr
SITEKEYFILE  = /etc/Tripwire/site.key
LOCALKEYFILE = /etc/Tripwire/$(HOSTNAME)-local.key
EDITOR      = /bin/vim
REPORTLEVEL  = 3
```

Этот файл нужно зашифровать и подписать. Сделайте это следующей командой:

```
# twadmin -m F -S site.key twcfg.txt
```

В результате будет создан файл **tw.cfg**. А во втором файле конфигурации, **/etc/Tripwire/twpol.txt**, как раз и содержатся все действия. Здесь вы устанавливаете правила, определяющие участки вашей системы, подлежащие мониторингу, и параметры проверки. Этот файл, в основном, подобен **Aide.conf**, рассмотренному ранее. В основном *Tripwire* поддерживает тот же набор атрибутов, что и *Aide*; они так похожи, что я не буду занимать место повторением, а просто отошлю вас к **Рис. 1**.

Как и *Aide*, *Tripwire* дает возможность задавать собственные наборы параметров (*Tripwire* называет их масками параметров) в виде комбинаций отдельных свойств файлов. Например, стандартный **twpol.txt** содержит следующие строки:

```
Growing    = +pinugtdl-srbamcCMSH ;
Device     = +pugsdr-intlbamcCMSH ;
Dynamic    = +pinugtd-srlbamcCMSH ;
IgnoreAll  = -pinugtsdrlbamcCMSH ;
IgnoreNone = +pinugtsdrbamcCMSH-l ;
ReadOnly   = +pinugtsdbmCM-rlacSH ;
Temporary  = +pugt ;
```

Первая строка задает маску параметров под названием Growing, с включением свойства pinugtdl и исключением srbamcCMSH.

Такая маска параметров подходит для постоянно растущих файлов (например, журналов) и примерно соответствует набору параметров Logs, рассмотренному нами в конфигурационном файле *Aide*.

Как и в *Aide*, файл политики *Tripwire* позволяет указать, какие маски параметров в каких участках файловой системы вы хотели бы применить – это и есть центральная часть конфигурации *Tripwire*. Вот несколько строк для примера:

```
/var/spool/mail -> $(Growing);
/bin            -> $(ReadOnly) ;
/lib            -> $(ReadOnly) ;
/sbin           -> $(ReadOnly) ;
/tmp i          -> $(Temporary);
!/proc;     # Ignore this directory
```

Первая строка предписывает *Tripwire* проверять все файлы из директории **/var/spool/mail** по маске Growing, которую мы только что обсуждали. Последняя строчка иллюстрирует применение символа ! для исключения указанных директорий. В отличие от *Aide*, *Tripwire* не поддерживает регулярных выражений в именах файлов. Я привел здесь лишь некоторые общие правила, а при вдумчиво построенной политике безопасности подобный файл может состоять из сотен строк.

Отредактировав файл политики, надо зашифровать и подписать его. Снова запустите *twadmin*, например вот так:

```
twadmin -m P twpol.txt
```

Вам будет предложено ввести пароль сайта. Программа сгенерирует зашифрованный файл **/etc/Tripwire/tw.pol**.

Устройство ловушки

Необходимая подготовка проделана, и мы можем создавать исходную базу данных о файловой системе. Сделайте это с помощью *# Tripwire --init*. Процесс займет несколько минут; если в **twpol.txt** попадутся несуществующие директории, во время работы вы увидите предупреждения. Предположим, вы воспользовались конфигурационным файлом, рассмотренным выше. Снимок системы будет создан в **/var/lib/Tripwire/hostname.twd** (замените **hostname** реальным именем машины). В отличие от *Aide*, просмотреть этот файл непосредственно вы не сможете, он двоичный.

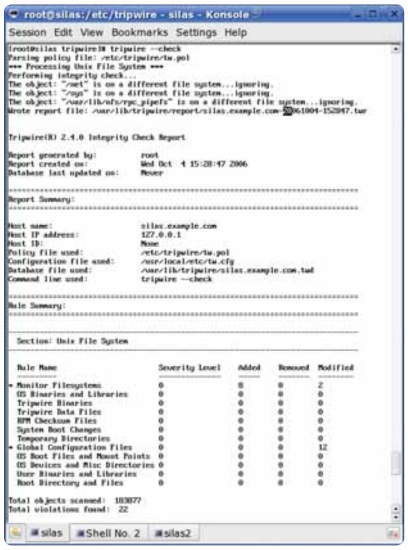
ОК, пора начинать атаку! Для этого наш воображаемый недоброжелатель просто добавит новую учетную запись (а может быть, зарегистрированный пользователь создаст ее для своего бойфренда) командой *# useradd barney*. Для запуска *Tripwire* в режиме проверки наберите команду *# Tripwire --check*. *Tripwire* пошлет все данные об изменениях на стандартный вывод, а также в файл с названием вроде **/var/lib/Tripwire/report/silas-20060915-152241.twr**, где «silas» – название машины (да, я тоже читал «Код да Винчи»), а строка цифр – дата и время. Отчет весьма обширен и занимает два экрана (**Рис. 4**). Взглянув на второй экранный снимок, можно увидеть, что изменились файлы **passwd**, **shadow** и **group** (кроме прочих), к тому же была создана домашняя директория Barney (с несколькими файлами конфигурации). И снова этот клятый **/etc/cups/cert/0**…

Просмотрев отчет *Tripwire*, вы можете решить, что некоторые изменения вполне законны: например, Barney – желанный гость. Если это так, обновим базу данных командой:

```
# cd /var/lib/Tripwire/report
# Tripwire --update -r silas-20060915-152241.twr
```

Для вас запустится текстовый редактор (тот, что указан в строке EDITOR файла конфигурации *Tripwire* или в переменной окружения EDITOR), и в нем откроется отчет *Tripwire*. Отметьте одобренные изменения. Выйдите из редактора, и база данных обновится, Это предотвратит отметку одних и тех же изменений при каждой проверке.

Tripwire и *Aide* – не единственные инструменты обнаружения вторжений. Среди заслуживающих внимания я отметил бы *Samhain*, *Osiris* и *Nabou*. Чтобы ознакомиться с ними, посетите **www.la-samhna.de/library/scanners.html**.



› (Рис. 4) Первая половина объемистого отчета *Tripwire*. Большинство отмеченных изменений вызвано добавлением учетной записи Barney.

Мысли напоследок

Итак, моя серия статей о безопасности подошла к концу. Быть может, вы видели впечатляющую пьесу театра *Reduced Shakespeare Company* под названием «*Полное собрание сочинений Шекспира (Адаптировано)*», где 27 сюжетов уместились в 97 минут. В подобной обработке (хотя и не столь артистично), для сообразительных и сильно занятых я привожу семь ключевых принципов, которые, надеюсь, вы усвоили в ходе наших занятий.

1. Применяйте сложные пароли, особенно для суперпользователя.
2. Избегайте прямого входа суперпользователя, когда им можно стать через *su*.
3. По возможности, не выдавайте пароль суперпользователя. При необходимости пользуйтесь *sudo* для контролируемого расширения прав доступа.
4. Отключите и/или удалите службы, которые вам не нужны.
5. Во время каждого обновления вашего дистрибутива убеждайтесь в том, что установлены новейшие доступные исправления безопасности.
6. Скрывайте компьютер за брандмауэром, а если не получается – создайте собственный фильтр пакетов для защиты машины.
7. Регулярно применяйте системы обнаружения вторжений типа *Tripwire* и *Aide*, и обязательно читайте их отчеты.

В этих принципах нет ничего сверхъестественного, но следование им существенно снизит риск для вашего компьютера. Желаю вам многолетней работы без взломов! LXF

Скорая помощь

Для поучительного чтения попробуйте «*Linux Server Security*» Майкла Бауэра [Michael Bauer], опубликованную O'Reilly. Превосходная книга. Помимо вопросов, затронутых в настоящей серии, охватывает множество других. Настоятельно рекомендую.

Ядро: Создай

Говорят, что звания линуксоида достоин только тот, кто лично собирал ядро. Вы готовы взяться за гуж? **Нейл Ботвик** покажет, как это сделать.





Наш эксперт

Нейл Ботвик – внештатный консультант по Linux, и ему претит готовое ПО, работающее «прямо из коробки». Поэтому он много лет собирает и ломает свои собственные ядра Linux.

Есть на свете фраза, с гарантией вселяющая ужас в сердца Linux-новобранцев, особенно тех, кто побаивается командной строки. Фраза проста: «Скомпилируйте свое ядро и...» Для непосвященных прохождение этого обряда едва ли не страшнее убийства льва или ритуальной скарификации. Запуск самодельного ядра отворяет дверь в королевство гордых линуксоидов.

Но что дает вам собственное ядро помимо законной гордости? Вот список возможных ответов:

- **Компактный объем.** Ядра, входящие в дистрибутивы, обычно поддерживают массу лишнего оборудования, о котором вы, возможно, сроду не слышали.
- **Дополнительные функции или драйвера.** Закон Мэрфи гласит: несмотря на предыдущее утверждение, в этой массе не найдется только одного драйвера – для вашего только что купленного устройства.
- **Ядро новее поставляемого с вашим дистрибутивом.** Вам хочется собрать его для поддержки нового оборудования, усиления безопасности – или просто потому, что душа просит.
- **Заплатки на ядро.** Существуют заплатки, пока не включенные в основную ветку ядра; некоторым из них там и не бывать. Если вы хотите насладиться каким-нибудь Reiser4, придется подлатать исходный код и скомпилировать ядро самим.

На этом уроке я покажу вам, что компиляция ядра не сильно отличается от компиляции большинства других пакетов: сначала идет запуск скрипта настройки, затем *make* для компиляции, а затем установка. Основное отличие – стадия настройки обычно интерактивная. Еще одно отличие – при каждой компиляции нового ядра оно устанавливается рядом с предыдущим, и в случае неудачи всегда можно загрузить старое, работающее ядро.

Для этого урока я воспользуюсь Mandriva Linux 2007, однако сам процесс одинаков для всех дистрибутивов. Мы будем работать только с ядрами серии 2.6; ядра серии 2.4 имеют некоторые отличия, но те, кто все еще работает с ними и ни разу не собирал ядро из исходных текстов, вряд ли когда-нибудь решат сделать это.

Обращение к истокам

Приступить к делу мы сможем не раньше, чем установим исходные тексты ядра. Для начала, воспользуемся исходными текстами, соответствующими вашему наличному ядру: ведь это готовая рабочая конфигурация, и ее можно принять за отправную точку. С помощью менеджера пакетов вашего дистрибутива установите нужные пакеты. Имя пакета варьируется от дистрибутива к дистрибутиву, но обычно оно выглядит как *kernel-source* или *linux-source*. В Mandriva 2007 ядро находится в пакете *kernel-2.6.17.5mdv-1-1mdv2007.0*, оно уже установлено, а исходные тексты – в пакете *kernel-source-2.6.17.5mdv-1-1mdv2007.0*, их надо будет установить из Центра Управления. Номер версии текущего ядра вам выведет команда **uname -r**.

Также понадобится установить компилятор и сопутствующие инструменты; установка *GCC* должна обеспечить установку всех остальных пакетов. Некоторые дистрибутивы имеют группы пакетов для разработки; если у вас стоит Ubuntu, то установка пакета *build-essentials*, подтянет другие пакеты в виде зависимостей.

Обычно исходные тексты Linux располагаются в каталоге **/usr/src/linux-номер-версии**, на который ссылается **/usr/src/linux**. Если ваш дистрибутив не создал ссылку, сделайте это сами, командами

```
cd /usr/src
ln -s linux-2.6.xyz linux
```

Так как мы используем существующую настройку как отправную точку, то должны убедиться, что в **/usr/src/linux** есть копия файла настройки. Файл называется **.config**; если его там нет, возможно, он в каталоге **/boot** или в пакете *linux headers*. Команда **locate .config** поможет его найти; скопируйте его в **/usr/src/linux**.

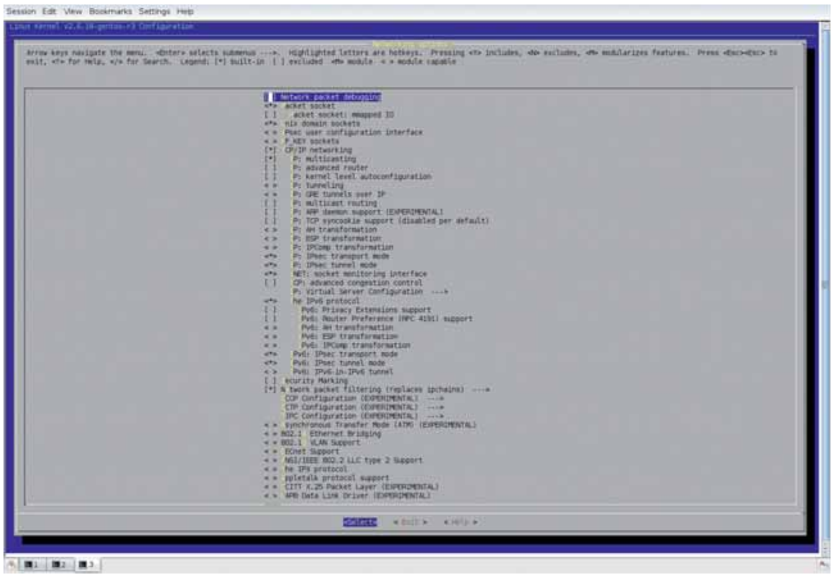
Установив исходные тексты ядра, можете начать его настройку с помощью следующих команд, выполняемых от лица root. Если вы

Вы — супер

Хотя латать ядро и настраивать и компилировать исходные тексты можно от лица обычного пользователя, для установки ядра необходимо стать суперпользователем [root]. Ради удобства предположим, что все команды запускаются в терминале от лица root. В некоторых дистрибутивах есть пункт меню для запуска этого терминала; в противном случае откройте терминал и наберите **su** -, после чего введите пароль суперпользователя. Ubuntu этого не позволяет, но тут поможет команда **sudo -i**: скажите ей ваш собственный пароль.

Месяц назад Пол Хадсон показал, как надо писать документацию в формате DocBook.

себе свое!



› Графический интерфейс на *Ncurses* скучноват, зато функционален и не требует *X*.

используете терминал в графическом режиме, будет удобнее раскрыть его на весь экран.

```
cd /usr/src/linux
make menuconfig
```

Теперь вы находитесь в инструменте настройки, использующем библиотеку *Ncurses*. Не пожалейте времени на осмотр и попробуйте различные варианты (перемещаясь с помощью курсорных клавиш). При выходе вас спросят, хотите ли вы сохранить настройку ядра; пока вы не ответили Да, ваши эксперименты ни на что не влияют. Большинство предлагаемых опций довольно очевидны. Любой пункт, оканчивающийся на --->, содержит подменю, куда вы можете войти, нажав Enter. Exit фактически переводит вас на уровень выше, и настоящий выход возможен только с самого верхнего уровня. Вместо Exit можно использовать клавишу Esc.

Обычно выбранные пункты могут иметь одну из трех отметок: *, M или ничего. * означает, что данный пункт будет встроен в ядро, M – что он будет скомпилирован как подгружаемый модуль ядра, а «ничего» означает, что опция отключена. Вы можете выбирать данные состояния с помощью клавиш Y, M и N соответственно, или переключаться между состояниями клавишей пробела. Еще одна клавиша, которая вам пригодится (особенно на первых порах) – ?, она выводит подсказку о текущем пункте.

Следует также знать о клавише /. В настройке ядра множество опций, разделенных на несколько уровней, поэтому поиск нужной именно вам может быть затруднен. Нажав /, вы сможете поискать то, что вам требуется.

Обрезка дерева ядра

Потратив некоторое время на знакомство с опциями (я вас не тороплю), попытаемся собрать новое ядро, удалив ненужные опции. Продолжайте нажимать Exit, пока вас не спросят о сохранении изменений. Скажите Нет и перезапустите *make menuconfig*. Сейчас заботливые разработчики дистрибутивов стараются встроить поддержку почти всего, чтобы их детище могло работать в самом широком диапазоне устройств. Поэтому займемся исключением избыточных драйверов.

Перейдите в раздел Device Drivers > Network Device Support > Ethernet (10 или 100Mbit) – увидите список всех драйверов сетевых карт, установленных в вашем ядре. Нужны вам, скорее всего, один или два, так что снимите «галочки» с остальных, которые не относятся к вашим устройствам. В случае затруднения команда *lsmod* покажет вам, какие модули загружены в данный момент, а файл помощи каждой опции поможет вам узнать имя модуля. Повторите этот процесс для разделов Ethernet (1000Mbit) и беспроводной сети [wireless LAN]. С удалением поддержки отсутствующих звуковых и видео карт торопиться не надо, этим мы займемся позже. Нажимайте Exit, пока не появится предложение сохранить файл настройки ядра – ответьте согласием.

Теперь пора скомпилировать и установить ваше новое ядро:

```
make all modules_install install
```

Сюда включены три шага: компиляция нового ядра и всех его модулей, установка модулей в **/lib/modules/версия-ядра** и, наконец, установка ядра в **/boot**. Прежде чем делать что-либо, проверьте, что загрузчик правильно настроен: на новое или на старое ядро. В **/boot** вы увидите, что *make install* поместил туда новое ядро.

В некоторых дистрибутивах *make install* способна на большее. Например, дистрибутивы на основе Debian, а также Gentoo и те, что используют скрипт Debian *installkernel*, создают еще и символьную ссылку: для нового ядра на **vmlinuz**, а для предыдущего на **vmlinuz.old**. Другие, как Mandriva, только копируют ядро и сопутствующие файлы. Чтобы поменьше печатать, я написал короткий скрипт для установки ссылок в таких дистрибутивах:

```
#!/bin/bash
cd /boot
NEW_KERNEL=$(ls -1rt vmlinuz-* | tail -n 1)
OLD_KERNEL=$(readlink vmlinuz)
NEW_VERSION=${NEW_KERNEL##vmlinuz-}
OLD_VERSION=${OLD_KERNEL##vmlinuz-}
for F in config System.map vmlinuz; do
  [ -f "${F}-${NEW_VERSION}" ] && ln -sf ${F}-${NEW_VERSION} ${F}
  [ -f "${F}-${OLD_VERSION}" ] && ln -sf ${F}-${OLD_VERSION} ${F}.old
done
[ -f "initrd-${NEW_VERSION}.img" ] && ln -sf initrd-${NEW_VERSION}.img initrd.img
[ -f "initrd-${OLD_VERSION}.img" ] && ln -sf initrd-${OLD_VERSION}.img initrd.img.old

[ -x /sbin/lilo ] && /sbin/lilo -v
```

Если вы отредактируете настройку загрузчика, вставив ссылки на **vmlinuz** и **vmlinuz.old**, то всегда сможете загрузить любое из ядер. При использовании *Grub* потребуется привести **/boot/grub/menu.lst** к следующему виду:

```
title Latest kernel
```

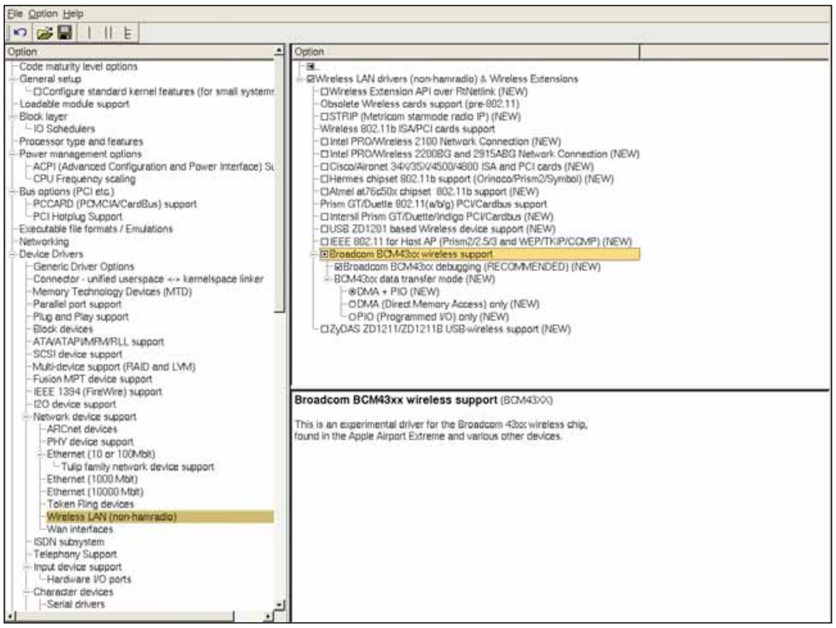
»

Скорая помощь

Есть одна опция, которую действительно стоит включить: General Setup > Enable Access to .config Through /proc/config.gz (IKCONFIG_PROC). Тогда вся настройка текущего ядра будет доступна в **/proc/config.gz**, и вы можете почитать о ней с помощью *zcat* или осуществить поиск через *zgrep*. Имея под рукой такую информацию, проще отследить, что именно вы натворили.

Скорая помощь

Некоторые дистрибутивы держат ядра в отдельном разделе /boot, который не обязательно иметь смонтированным в работающей системе: это защищает ядро от повреждения. Если ваша система именно такова, убедитесь, что вы смонтировали /boot до установки нового ядра, не то при загрузке попадете в старое.



› **Можете взять графический инструмент настройки, например, *Make Xconfig*.**

```
kernel /vmlinuz root=абв... и другие опции загрузки
title Previous kernel
kernel /vmlinuz.old root=абв... другие опции загрузки
```

Если вы используете *Lilo*, то должны сделать аналогичные изменения в настройке и не забыть запустить *Lilo* после установки нового ядра. Скрипт Debian *installscript* и скрипт, приведенный выше, позаботятся об этом за вас.

Осторожно, ловушки!

Можно значительно уменьшить размер ядра, убрав ненужный код, тем самым уменьшив занимаемую им память, а также время загрузки. Будьте как дома, но не переборщите. Удаляйте только действительно ненужные опции. Помните, что ваше предыдущее ядро сохраняется: если что-то пойдет не так и вы не сможете загрузиться, выберите старое ядро в загрузочном меню.

Есть пара ловушек для неосмотрительных, которые не позволяют системе загрузиться. При выборе файловой системы проверьте, что файловая система, используемая корневым разделом, встроена в ядро, а не является его модулем (подробности см. ниже во врезке Драйверы: модули или встроенные?). Также проверьте, что вы включили поддержку вашего чипсета IDE, SATA или SCSI; они тоже должны быть встроены. Ядра, входящие в дистрибутивы, вполне могут компилировать критические участки кода как модули, потому что загружают их с загрузочного диска (**initrd**), размещаемого в оперативной памяти [ramdisk]. Ваш дистрибутив, возможно, использует файл **initrd** из **/boot** – его необходимо указать в настройках загрузчика.

Initrd необходим при изготовлении дистрибутивов общего назначения. Как я уже сказал, некоторые компоненты, например, драйверы вашего дискового контроллера, нельзя компилировать как модули, потому что модули не будут доступны, пока не смонтирован корневой раздел файловой системы. Но если каждую возможную опцию компилировать в ядро, оно станет неподъемным. Поэтому модули помещаются на ramdisk, который система изначально монтирует как корневой раздел. Затем стартовый сценарий подгружает с него необходимые модули, монтирует настоящий корневой раздел и передает управление.

При сборке сугубо личного ядра нужды в этом нет – все необходимые драйверы умещаются в ядре (у вас ведь только один контроллер диска и одна файловая система), а оставшиеся 99% опций можно выкинуть. Единственно, когда вам может понадобиться initrd при использовании самодельного ядра – это если вы захотите поумничать: управлять файловой системой через LVM или выводить чудо-экран во время загрузки системы.

Вернемся к нашей задаче. В этот раз мы использовали *make menuconfig* для настройки ядра. Преимущество этого способа в том, что он не использует X, поэтому вы можете использовать его даже удаленно через SSH. Но если вы предпочитаете графику, то существует парочка графических оболочек: *Make xconfig* и *Make gconfig*. Первая использует оболочку на основе *Qt*, вторая – на основе *GTK*. Они предоставляют многопанельный интерфейс, в котором одна панель отображает описание для текущей опции – полезно, когда вы просматриваете ядро в поисках того, что можете сделать. Альтернатива *Qt* также содержит опцию поиска, но она не так полезна, как в *menuconfig*, которая показывает вам, где расположена опция и что еще вам необходимо включить. Вы можете выбрать из трех альтернатив. Тяжело выбрать фаворита, поэтому попробуйте их все.

А вдруг вы решите взять исходные тексты ядра из других источников? Хороший вопрос. Пусть, например, поддержка вашего оборудования появилась только в последнем релиз-кандидате. Вы скачиваете архив исходных текстов ядра с www.kernel.org (пререлизы находятся в www.kernel.org/pub/linux/kernel/v2.6/testing), распаковываете его в **/usr/src**, меняете символьную ссылку **linux** способом, описанным выше, и копируете файл **.config** вашего предыдущего ядра в этот каталог.

Теперь вы понимаете, почему мы начали с исходных текстов вашего существующего ядра. Хотя вы и работаете с другой версией, большинство настроек следует сохранить. А если понадобятся новые настройки, как их найти и отрегулировать? Для этого существуют специальные средства. Сначала перейдите с помощью *cd* в **/usr/src/linux** и наберите

```
make oldconfig
```

Команда пройдется по вашему файлу настройки и предъявит вам все новые, еще не настроенные опции. Как обычно, для каждой опции вас попросят выбрать Y/M/N/?. Если сомневаетесь, жмите ?, и все о ней узнаете. Проделав все это, можете набрать *make menuconfig/xconfig* и

Драйверы: модули или встроенные?

Ядро Linux монолитное – изначально подразумевалось, что все драйверы будут частью ядра. Но ядро росло и стало громоздким, поэтому была предложена концепция загружаемых модулей. Они существуют как файлы на диске и загружаются в ядро по мере необходимости. Это обеспечивает необычайную гибкость – а также бесконечные споры о том, встраивать ли драйвера в ядро или делать отдельные модули. Ответ зависит от вашего компьютера и от того, как он будет использоваться.

К примеру, у моего ноутбука два сетевых интерфейса, проводной и беспроводной. Я могу использовать как оба, так и ни одного, поэтому имеет смысл использовать модули. С другой стороны, мой настольный компьютер имеет один, проводной интерфейс, который всегда при деле – почему бы и не встроить его драйвер в ядро, он всегда будет загружен.

Работа вообще без модулей и запрещение их загрузки в ядро повышает безопасность системы. Это хороший вариант для сервера, подключенного к Интернету: серверу требуется не так уж много драйверов, и лучше их встроить. Модульное ядро больше подходит настольным компьютерам и ноутбукам, потому что им приходится работать со всевозможными периферийными устройствами.

Популярные наборы

Вот некоторые наборы, с которыми вы можете захотеть поэкспериментировать:

» ***mm*** Заплатки Эндрю Мортона на www.kernel.org.

» ***ck*** Проживающие по адресу http://members.optusnet.com.au/ckolivas/kernel, заплатки Кона Коливаса [Con Colivas] сделают ваш рабочий стол быстрее и отзывчивее.

» ***viper*** Экспериментальные заплатки, призванные уменьшить латентность, накладные расходы и увеличить интерактивность. См. http://vipernicus.evolution-mission.org.

» ***no*** Набор заплаток *mm* плюс ряд других, увеличивающих производительность и гибкость. Получите его на http://no.oldos.org.

подкорректировать параметры, а потом уж наберите *make all modules_install install*; ядро скомпилируется и установится как и ранее.

Должен предупредить вас о потенциальной проблеме, возникающей при компиляции ядра, взятого с Kernel.org: многие дистрибутивы в той или иной степени модифицируют ядро для своих целей, и при установке чистого ядра некоторые вещи могут отказаться работать, не найдя необходимых заплаток. Пользователям Slackware об этом беспокоиться нечего, но владельцы любого другого дистрибутива должны предвидеть подобную возможность.

Латаем ядро

Мы уже почти закончили: по-моему, я рассказал о большинстве доводов для сборки своего ядра: уменьшение объема, включение новых возможностей и использование нового ядра. А как насчет заплаток на ядро? Да не убегайте, вернитесь! Не так все сложно. Предположим, что вы скачали нужную заплатку на ваше ядро, ну хоть Reiser4 для ядра 2.6.18.3 – поставьте ее следующим образом:

```
cd /usr/src/linux
gunzip reiser4-for-2.6.18-3.patch.gz
patch -p1 <reiser4-for-2.6.18-3.patch
```

Здесь подразумевается, что заплатки сжаты с помощью *gzip* (обычный вариант); ну, а если они сжаты с помощью *bzip2*, используйте bunzip2. Вы можете пропустить этап распаковки, если замените последние две строчки на эту (для заплатки, упакованной с помощью *bzip2*, используйте bzcat):

```
zcat reiser4-for-2.6.18-3.patch.gz | patch -p1
```

Вы увидите список измененных файлов – надеюсь, в нем не будет сообщений об ошибках. Ошибки обычно означают, что вы взяли заплатку, не соответствующую версии вашего ядра.

Применив заплатку, запустите *make menuconfig/xconfig/oldconfig* чтобы установить новые опции, затем запустите *make all modules_install install*. Если заплатка не срабатывает, вам может помочь изменение значения параметра -p. Взгляните в начало файла-заплатки: вы увидите нечто похожее на

```
--- linux-2.6.18.orig/arch/i386/lib/usercopy.c
+++ linux-2.6.18/arch/i386/lib/usercopy.c
```

Здесь указано имя файла, который будет изменен, а также сами изменения. Имя файла прописано относительно текущего каталога. Опция -p для патча определяет число ведущих каталогов (строго говоря, число слэшей) для удаления. Мы уже находимся в каталоге **linux-2.6.18**, поэтому -p1 удаляет эту часть пути и ищет файл **arch/i386/lib/usercopy.c**. Если бы мы были в каталоге **/usr/src**, нам понадобилось бы использовать опцию -p0, но в нашей конфигурации она не работает, потому что исходный каталог называется **linux-2.6.18.3**, а не **linux-2.6.18**. Применение заплатки из каталога **/usr/src/linux** обычно является наиболее безопасным выбором.

Все это может показаться ерундой, но при установке заплаток подбор верного значения -p часто оказывается сложнее всего прочего. Однако есть опция --dry-run, с которой вы можете начать. Она просто проверяет, корректно ли наложатся заплатки, не внося никаких изменений.

Одну заплатку наложить на ядро таким способом довольно просто. А вот в случае их множества могут возникнуть проблемы, так как разные заплатки могут менять одни и те же файлы. Тогда все зависит от порядка их наложения. Множество заплаток порождает множество опций. Можете посидеть в Google, или почитать рассылку Linux Kernel Mailing List, или придумать еще что-нибудь, чтобы заставить заплатки работать совместно. А можете применить коллекцию заплаток. Такие коллекции добавляют множество возможностей и расширений одной командой patch, и их довольно много: от уважаемых и стабильных до крайне рисковых. В первой группе прочно обосновался набор заплаток *mm*.

MM, заплатки!

Набор заплаток *mm* подготовлен Эндрю Мортоном [Andrew Morton], хранителем ядра серии 2.6. Набор доступен на сайте www.kernel.org, и официальнее, чем он, неофициальному набору и быть нельзя. Он содержит заплатки, которые могут быть внесены в официальную ветку ядра 2.6 в будущем, но пока считаются недостаточно протестированными для повсеместного применения. В наборе *mm* изменено более 3 850 файлов (из почти 21 000 файлов, входящих в ядро) – можете представить, сколько работы потребуется для применения заплаток по отдельности.

Применение заплатки осуществляется обычным образом. Для текущего последнего набора надо сделать следующее:

```
cd /usr/src/linux-2.16.19-rc6
bzcat /path/to/2.6.19-rc6-mm1 | patch -p1
```

Затем настройте, исследуйте новые опции, скомпилируйте и установите. Теперь вы гордый владелец крутого, «пропатченного» ядра. Как видите, ничего страшного – но не обязательно этого рассказывать восхищенным новичкам Linux! LXF

Сторонняя компиляция

Даже если вы не собираетесь создавать собственное ядро, может случиться, что вам понадобятся исходные тексты ядра в **/usr/src/linux**. Если вы хотите установить приложение, добавляющее собственные модули ядра, то ему может понадобиться доступ к исходным текстам ядра и компилятор для создания этих модулей. Типичные примеры – драйвера Nvidia и система эмуляции *VMware*. И те, и другие требуют специальных модулей ядра. Они содержат ряд предварительно скомпилированных модулей для популярных дистрибутивов и версий ядра, но если вашей системы нет в списке или вы обновились, то их установщики должны будут собрать новые модули.

Отсюда следует, что, обновив ядро, хотя бы и в той же версии, придется переустановить все пакеты, которые добавляли свои собственные модули.

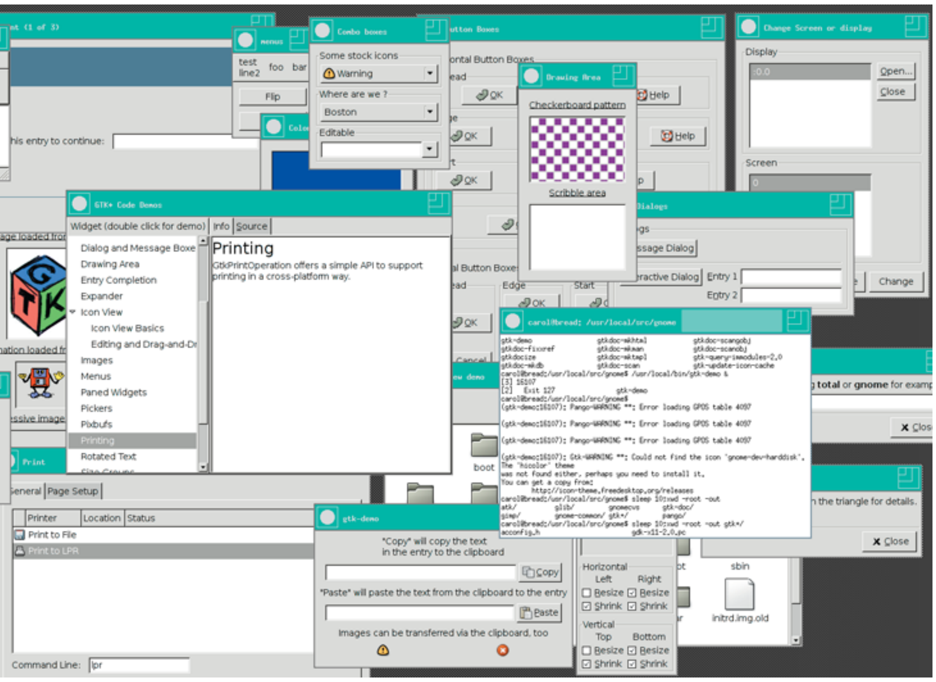
Скорая помощь

Есть несколько способов убрать за собой, особенно при значительных изменениях в конфигурации. Прежде всего, *make clean* удаляет все скомпилированные файлы из дерева ядра, возвращая вас практически в состояние до компиляции. Для более тщательной уборки используйте *make mrproper*, чтобы прочистить также все файлы настроек. Перед этим, однако, разумно будет создать резервную копию файла **.config**.

» **Через месяц** Как настроить загрузчик системы на примере *Grub*.

Изучаем сигналы

ЧАСТЬ 3: Наше знакомство с сигналами и событиями *GTK+* было прервано изучением увлекательной проблемы интернационализации приложений. Сегодня **Андрей Боровский** намерен вернуться к ним и показать, что их не зря называют движущей силой любой программы, основанной на *GTK+*.



Вы, конечно, заметили, что в самом первом примере программы *GTK+* мы использовали две разные функции для связывания обработчика сигнала и объекта: g_signal_connect() и g_signal_connect_swapped(). Общим у этих функций является то, что в их первом параметре передается объект-источник сигнала, для которого назначается обработчик. Разница между этими функциями заключается в порядке передачи аргументов функции-обработчику сигнала. Если обработчик сигнала был связан с объектом с помощью g_signal_connect(), первым параметром, передаваемым обработчику, является первый параметр g_signal_connect(), то есть указатель на объект-источник сигнала (информация об объекте-источнике может быть полезной, поскольку один обработчик сигналов может быть связан с несколькими объектами). Последним параметром, который получает обработчик сигнала, связанного с помощью g_signal_connect(), является последний параметр g_signal_connect(), в котором обработчику передаются произвольные дополнительные данные. Если объект и сигнал были связаны между собой с помощью g_signal_connect_swapped(), при вызове обработчика сигнала первым параметром, передаваемым обработчику, является последний параметр g_signal_connect_swapped() (что бы он там ни содержал). В то же время, первый параметр g_signal_connect_swapped(), который представляет собой указатель на объект-источник сигнала, становится последним параметром, передаваемым обработчику сигнала. Зачем же нужны две разные функции связывания объекта и обработчика сигнала?

Для связывания наших собственных обработчиков сигнала мы будем использовать, в основном, функцию g_signal_connect(), так что заголовок функции-обработчика сигнала будет иметь вид:

```
void callback_func( GtkWidget *widget,
            ... /* дополнительные аргументы, в зависимости от типа сигнала */
            gpointer   callback_data );
```

где widget – первый параметр g_signal_connect(), а callback_data – последний параметр.

Функция g_signal_connect_swapped() используется в двух ситуациях. Во-первых, ее применяют, когда в качестве обработчика сигнала, эмитируемого объектом, нужно назначить функцию, которая, вообще говоря, не является обработчиком сигнала. Так, например, мы поступали при вызове

```
g_signal_connect_swapped(G_OBJECT(button1), "clicked", G_CALLBACK(gtk_widget_destroy), G_OBJECT(window));
```

Функция gtk_widget_destroy() не является обработчиком какого-либо сигнала (чтобы подчеркнуть это, мы явным образом приводили gtk_widget_destroy к типу GtkCallback), эта функция просто уничтожает визуальный элемент, указатель на который передан ей в качестве аргумента. Наша задача – в ответ на щелчок кнопки вызывать gtk_widget_destroy() для объекта window.

Если бы мы связывали сигнал clicked с функцией gtk_widget_destroy() с помощью функции g_signal_connect(), аргументом функции gtk_widget_destroy() стал бы объект button1. Однако, поскольку мы используем функцию g_signal_connect_swapped(), первым аргументом, переданным функции gtk_widget_destroy(), окажется последний аргумент g_signal_connect_swapped(), то есть, объект window. Я надеюсь, что вы понимаете, почему вместо функции g_signal_connect_swapped() нельзя использовать функцию g_signal_connect() с переставленными местами первым и последним аргументами. Ведь в этом случае сигнал окажется связанным не с тем объектом!

В примере buttontest.c, который вы найдете на диске, мы связываем сигнал clicked кнопки button2 с функцией g_print():

```
g_signal_connect_swapped(G_OBJECT(button2), "clicked", G_CALLBACK(g_print), "Button is pressed!\n");
```

Как вы уже знаете, последний аргумент g_signal_connect_swapped() станет первым аргументом функции g_print(). В результате, каждый раз при щелчке по кнопке, на экране терминала будет распечатываться строка "Button is pressed!". Указатель на объект button2 передается функции g_print() как второй аргумент. В приведенном выше фрагменте

Месяц назад Мы изучали процесс интернационализации приложений *GTK+*.

и события

Linux Format на DVD
» Код примеров статьи

Передача аргументов функциям С

Современные компиляторы С следят за тем, чтобы число (и тип) аргументов, передаваемых функции при ее вызове, совпадали с числом и типом аргументов, указанным в заголовке функции, однако, если функция вызывается косвенно (как, например, функция-обработчик сигнала), в силу вступает старое правило С: обрабатываются первые **n** аргументов, на которые есть ссылки в теле функции, а остальные аргументы просто игнорируются. Например, при вызове функции gtk_widget_destroy() как обработчика события clicked, функции передается два аргумента. Первый аргумент функция использует, второй – игнорирует.

кода функция g_print() игнорирует все переданные ей аргументы, кроме первого, но если вы замените последний аргумент g_signal_connect_swapped() на "Button is Pressed %i\n", то увидите, что кроме текста на экране будет распечатано число, являющееся численным представлением указателя на объект button2.

Еще одна ситуация, в которой мы можем обратиться к функции g_signal_connect_swapped(), возникает тогда, когда мы хотим, чтобы функция-обработчик, вызванная для обработки события, источником которого является один объект, думала, что обрабатывает событие, связанное с другим объектом. Если вы считаете, что обманывать обработчики событий нехорошо, то в большинстве случаев вы правы. Однако, иногда такой «обман» может оказаться полезным, как будет показано в следующем примере.

События

Мы уже упоминали, что события *GTK+* представляют собой разновидность сигналов. В отличие от остальных сигналов *GTK+*, события тесно связаны с событиями системы X-Window. С одним из событий, а именно – delete_event, мы уже встречались. В общем виде заголовок функции-обработчика события выглядит так:

```
gint callback_func(GtkWidget *widget,
            GdkEvent  *event,
            gpointer  callback_data );
```

Здесь необходимо сделать некоторые уточнения. Так же как объект widget является корнем иерархии объектов, представляющих различные визуальные элементы и, зачастую, нам приходится приводить тип GtkWidget * к типу указателя на соответствующий визуальный элемент, тип GdkEvent является корнем иерархии объектов событий, в которой каждому событию соответствует свой объект. Объект события – это обычная структура С, которая несет о нем специфическую информацию (поскольку разные события несут разную информацию, неудивительно, что каждому событию соответствует своя структура). Например, событию button_press_event, возникающему при щелчке мышью, соответствует объект GdkEventButton. Он представляет собой структуру С, которая, помимо прочего, имеет поле button (содержит информацию о том, какая кнопка была нажата), и поля x и y, содержащие координаты указателя мыши в момент щелчка.

В качестве примера использования событий мы рассмотрим программу *watchwindow*, которая следит за состоянием своего главного окна (распахнуто на весь экран, свернуто на панель задач, занимает часть экрана). Ниже приводится исходный текст программы (вы найдете его также в файле **watchwindow.c** на диске).

```
#include <gtk/gtk.h>
gint on_window_state(GtkLabel * label, GdkEventWindowState * window_
state, gpointer callback_data )
{
 if (window_state->new_window_state & GDK_WINDOW_STATE_
ICONIFIED) {
  g_print("Window is iconified\n");
  return 0;
 }
 if (window_state->new_window_state & GDK_WINDOW_STATE_
MAXIMIZED) {
  gtk_label_set_text(label, "Window is maximized");
  return 0;
 }
 gtk_label_set_text(label, "Window is in normal state");
 return 0;
}
static gboolean delete_event(GtkWidget * widget, GdkEvent * event,
gpointer data)
{
 return FALSE;
}
static void destroy(GtkWidget * widget, gpointer data)
{
 gtk_main_quit();
}

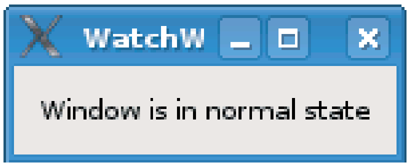
int main(int argc, char ** argv)
{
 GtkWidget * window;
 GtkWidget * label;
 gtk_init(&argc, &argv);
 window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
 gtk_window_set_title(GTK_WINDOW(window), "WatchWindow");
 gtk_container_set_border_width(GTK_CONTAINER(window), 10);
 g_signal_connect(G_OBJECT(window), "delete_event", G_
CALLBACK(delete_event), NULL);
 g_signal_connect(G_OBJECT(window), "destroy", G_CALLBACK(destroy),
NULL);
 label = gtk_label_new("Starting");
 gtk_container_add(GTK_CONTAINER(window), label);
 g_signal_connect_swapped(G_OBJECT(window), "window_state_event",
G_CALLBACK(on_window_state), G_OBJECT(label));
 gtk_widget_show(label);
 gtk_widget_show(window);
 gtk_main();
 return 0;
}
```

Главным визуальным элементом главного окна программы является объект label типа GtkLabel. Этот объект представляет собой простую текстовую метку, на которую выводится текст о текущем состоянии окна. Для того, чтобы главное окно могло получать информацию об изме- »

» нении своего состояния, оно должно обрабатывать сигнал-событие window_state_event. Мы связываем обработчик этого события, функцию on_window_state(), с объектом-источником события window с помощью функции g_signal_connect_swapped() Это как раз тот случай, когда применение g_signal_connect_swapped() оправдано. В качестве последнего аргумента мы передаем функции g_signal_connect_swapped() указатель на объект label. Из этого следует, что функция-обработчик сигнала получит в качестве первого аргумента указатель на объект label, а не на объект window.

Перейдем теперь к функции-обработчику. Поскольку мы знаем, какое именно событие будет обрабатывать эта функция, мы можем заменить базовые типы аргументов GtkWidget * и GdkEvent * в ее заголовке на те типы, с которыми функции в действительности придется иметь дело, то есть на GtkLabel * и GdkEventWindowState *. Это позволит нам сократить исходный текст программы на пару строк за счет преобразования типов. Структура GdkEventWindowState, которую можно условно рассматривать как потомок объекта GdkEvent, несет информацию о событии window_state_event. Информация о состоянии окна содержится в поле new_window_state структуры GdkEventWindowState. Это поле содержит набор флагов, отражающих новое состояние кона. В начале функции обработчика мы проверяем, установлен ли в поле new_window_state флаг GDK_WINDOW_STATE_ICONIFIED. Наличие этого флага означает, что окно свернуто на панель задач. В этом случае мы с помощью функции g_printf() распечатываем в окне терминала строку "Window is iconified" (как вы понимаете, нет смысла выводить что-либо в поле окна, которое свернуто на панель задач) и завершаем работу функции. Если флаг не установлен, мы проверяем, стоит ли флаг GDK_WINDOW_STATE_MAXIMIZED (окно распахнуто на весь экран). Если да, то мы выводим в поле метки label текст "Window is maximized" и завершаем выполнение функции-обработчика. Если же ни тот, ни другой флаги не установлены, мы задаем в качестве текста метки строку "Window is in normal state".

Порядок проверки флагов new_window_state имеет значение, поскольку, как это ни странно, флаги GDK_WINDOW_STATE_ICONIFIED и GDK_WINDOW_STATE_MAXIMIZED могут быть установлены одновременно. Такое происходит, когда окно сворачивается на панель задач из максимально распахнутого состояния. Поэтому сначала мы проверяем, установлен ли флаг GDK_WINDOW_STATE_ICONIFIED (что точно свидетельствует о том, что окно свернуто), и только если этот флаг не установлен, проверяем, установлен ли флаг, указывающий на то, что окну приданы максимальные размеры. Поле new_window_state может содержать и другие флаги, но они нас сейчас не интересуют. Таким образом окно нашего приложения способно получить (и сообщить нам) важную информацию о своих размерах (**рис. 1**).



› **Рис. 1. Окно, которое знает о своем состоянии.**

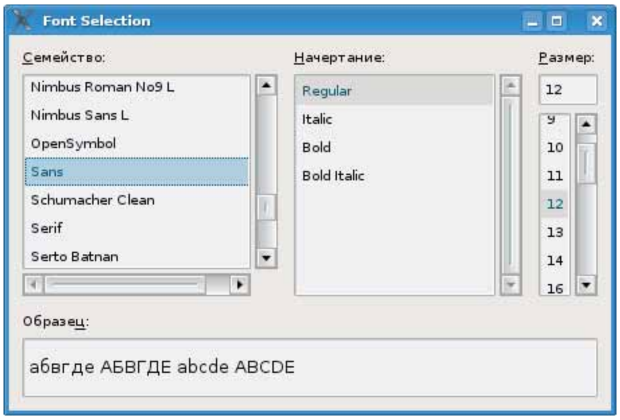
Продолжаем изучать контейнеры

Ранее мы уже имели дело с объектами-контейнерами GtkHBox и GtkVBox. Прежде чем переходить к сложным средствам компоновки, нам стоит познакомиться и с другими типами контейнеров.

В серии, посвященной *Qt*/KDE, я приводил пример приложения, предназначенного для просмотра шрифтов. Не могу удержаться от того, чтобы не привести пример подобного приложения для *GTK+*, тем более что написать его совсем несложно (исходный текст программы вы найдете в файле **fontview.c**):

```
#include <gtk/gtk.h>
static gboolean delete_event(GtkWidget * widget, GdkEvent * event,
gpointer data)
{
 return FALSE;
}
static void destroy(GtkWidget * widget, gpointer data)
{
 gtk_main_quit();
}
int main(int argc, char ** argv)
{
 GtkWidget * window;
 GtkWidget * font_selection;
 gtk_init(&argc, &argv);
 window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
 gtk_window_set_title(GTK_WINDOW(window), "Font Selection");
 gtk_container_set_border_width(GTK_CONTAINER(window), 10);
 g_signal_connect(G_OBJECT(window), "delete_event", G_
CALLBACK(delete_event), NULL);
 g_signal_connect(G_OBJECT(window), "destroy", G_CALLBACK(destroy),
NULL);
 font_selection = gtk_font_selection_new ();
 gtk_container_add(GTK_CONTAINER(window), font_selection);
 gtk_widget_show(font_selection);
 gtk_widget_show(window);
 gtk_main();
 return 0;
}
```

Основным элементом главного окна этой программы является объект GtkFontSelection, который представляет собой полноценный визуальный элемент просмотра и выбора шрифтов (**рис. 2**).



› **(Рис. 2) Компонент выбора шрифтов в окне приложения.**

Согласно принципам *GTK+*, объект GtkFontSelection создается функцией gtk_font_selection_new(). Мы связываем объект с окном с помощью функции gtk_container_add() и делаем его видимым при помощи вызова gtk_widget_show(). На самом деле, объект GtkFontSelection предназначен, конечно, не для построения программ просмотра наличествующих в системе шрифтов. Назначение этого объекта – быть частью окон настроек свойств приложений *GTK+*, допускающих выбор шрифта для вывода различных текстовых элементов.

Сейчас для нас важно, что объект GtkFontSelection является прямым потомком объекта GtkVBox. Ничего странного тут нет, поскольку, в принципе, любой сложный визуальный элемент, содержащий несколько дочерних элементов, должен быть потомком какого-либо контейнера. Хотите узнать, какие визуальные компоненты содержит контейнер GtkFontSelection, и даже получить к ним доступ? Это просто. После того как объект font_selection создан, добавьте в текст программы строку:

```
gtk_container_foreach(GTK_CONTAINER(font_selection), child_callback,
NULL);
```

Функция gtk_container_foreach() вызовет функцию обратного вызова, в данном случае – child_callback(), для каждого непосредственного дочернего визуального элемента контейнера font_selection. Последний параметр gtk_container_foreach() предназначен для передачи произвольных данных. Функция child_callback() выглядит так:

```
void child_callback(GtkWidget * widget, gpointer data)
{
 g_print("%s\n", gtk_widget_get_name(widget));
 if (GTK_IS_CONTAINER(widget))
```

```
  gtk_container_foreach (GTK_CONTAINER(widget), child_callback,
NULL);
}
```

Первый параметр функции – очередной дочерний визуальный элемент, для которого она вызвана, второй параметр – дополнительные данные. Мы распечатываем имя переданного нам визуального элемента, затем проверяем, является ли он контейнером. Если переданный нам виджет сам является контейнером, мы рекурсивно вызываем для него функцию gtk_container_foreach(). В результате выполнения программы будет распечатан список GtkTable, GtkScrolledWindow, GtkTreeView, GtkScrolledWindow, GtkTreeView, GtkScrolledWindow, GtkTreeView, GtkLabel, GtkLabel, GtkLabel, GtkEntry, GtkVBox, GtkLabel, GtkHBox, GtkEntry. Этот список представляет собой нечто вроде рекурсивного обхода дерева дочерних визуальных элементов контейнера в порядке 1-2-3. Функцию Отметим, что gtk_container_foreach() можно использовать только для перечисления внутренних элементов контейнера, то есть таких, которые не были добавлены явным образом.

Перейдем теперь к еще одному базовому типу контейнера, который должен особенно понравиться тем, кто программировал на Delphi, Borland C++ Builder или с Windows.Forms. Контейнеры GtkHBox и GtkVBox тают огромные возможности в плане компоновки элементов интерфейса. Такие приложения, как текстовый редактор, web-браузер или утилита настройки оборудования можно написать, используя для компоновки визуальных элементов исключительно «горизонтальный» и «вертикальный» контейнеры. Полезная особенность этих типов контейнеров заключается в том, что по умолчанию они управляют расположением и размером дочерних элементов наиболее естественным образом, так что пользователю не приходится беспокоиться о дополнительных настройках. Тем не менее, свободы, предоставляемой этими контейнерами, не всегда достаточно. Если вы пишете игровое или мультимедиа-приложение, вы можете захотеть расположить визуальные элементы управления совершенно необычным образом. Для этого вам следует воспользоваться контейнером GtkFixed. Объект-контейнер GtkFixed является прямым потомком объекта GtkContainer, от которого происходит также объект GtkBox, являющийся родоначальником объектов GtkHBox и GtkVBox. Этот контейнер позволяет располагать дочерние элементы в фиксированных позициях, заданных координатами x и y относительно верхнего левого угла контейнера. Контейнер GtkFixed является «фиксированным» потому, что, в отличие от «горизонтального» и «вертикального» контейнеров, расположение и размеры его дочерних элементов не меняются при изменении размеров окна. Он дает вам большую свободу в расположении дочерних элементов, но требует и большей ответственности при управлении ими. Рассмотрим простую программу, использующую контейнер GtkFixed (исходный текст вы найдете в файле **fixedbuttons.c**):

```
#include <gtk/gtk.h>
static gboolean delete_event(GtkWidget * widget, GdkEvent * event,
gpointer data)
{
  return FALSE;
}
static void destroy(GtkWidget * widget, gpointer data)
{
  gtk_main_quit();
}
int main(int argc, char ** argv)
{
  GtkWidget * window;
  GtkWidget * fixed_container;
  GtkWidget * button;
  gtk_init(&argc, &argv);
  window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  gtk_window_set_title(GTK_WINDOW(window), "Fixed Buttons Demo");
  gtk_container_set_border_width(GTK_CONTAINER(window), 10);
  g_signal_connect(G_OBJECT(window), "delete_event", G_
CALLBACK(delete_event), NULL);
  g_signal_connect(G_OBJECT(window), "destroy", G_CALLBACK(destroy),
NULL);
  fixed_container = gtk_fixed_new ();
  gtk_container_add(GTK_CONTAINER(window), fixed_container);
  button = gtk_button_new_with_label("Button1");
  gtk_fixed_put(GTK_FIXED(fixed_container), button, 5, 5);
     gtk_widget_show(button);
  button = gtk_button_new_with_label("Button2");
  gtk_fixed_put(GTK_FIXED(fixed_container), button, 25, 35);
  gtk_widget_show(button);
  button = gtk_button_new_with_label("Button3");
  gtk_fixed_put(GTK_FIXED(fixed_container), button, 45, 65);
  gtk_widget_show(button);
  gtk_widget_show(fixed_container);
  gtk_widget_show(window);
  gtk_main();
  return 0;
}
```

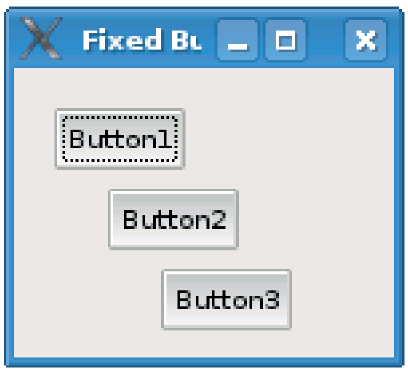
Мы создаем объект GtkFixed с помощью функции gtk_fixed_new(). Далее мы последовательно создаем и добавляем в контейнер три кнопки. Добавление нового элемента в контейнер выполняется функцией gtk_fixed_put(). Первым аргументом этой функции должен быть, естественно, указатель на объект-контейнер. Вторым аргументом является указатель на добавляемый в контейнер дочерний объект, а третий и четвертый аргументы служат, соответственно, для передачи координат x и y верхнего левого угла дочернего объекта. Каждый дочерний элемент, а также сам контейнер, необходимо сделать видимым с помощью вызова gtk_widget_show().Теперь кнопки в нашем приложении расположены по диагонали (**рис. 3**). Расположение уже добавленных в контейнер элементов можно изменять с помощью функции gtk_fixed_move(). Список аргументов у этой функции такой же, как и у gtk_fixed_put().

У объекта GtkFixed есть свойство children, которое содержит список всех дочерних элементов. Список представляет собой объект с не очень благозвучным для русского уха названием GList (этот объект реализует в *GTK+* классический связный список). Элементами списка children являются объекты типа GtkFixedChild. Объект GtkFixedChild содержат указатель на соответствующий ему дочерний объект и его координаты. Ниже показано, как с помощью списка children можно вывести на терминал координаты всех дочерних объектов контейнера GtkFixed.

```
GList * list;
GtkFixedChild * child;
...
list = GTK_FIXED(fixed_container)->children;
while (list != NULL) {
  child = list->data;
  g_print("x = %i, y = %i\n", child->x, child->y);
  list = g_list_next(list);
}
```

Поскольку GList представляет собой классический связный список, макросы типа g_list_next() и g_list_previous() возвращают указатель на элемент GList.

На этом наше, в чем-то – шапочное, знакомство с объектами-контейнерами можно считать законченным. Однако, как вы, наверное, помните еще со времен серии о Qt/KDE, пользоваться всем этим богатством напрямую зачастую оказывается неудобно. В следующий раз мы узнаем, как конструировать пользовательские интерфейсы в режиме WYSIWYG при помощи Glade.



› **Рис. 3. Произвольное расположение окон.**

» Через месяц Визуальное проектирование интерфейсов в Glade.

LINUX FORMAT на DVD
» Код примеров статьи

Демоны

ЧАСТЬ 8: Подобно леммингам, бесконтрольно размножающиеся потоки очень быстро устраивают гонку-соревнование за ресурсы системы. Но **Андрей Боровский** знает, как умерить их аппетиты...

-Я и есть демон! Слушай, малыш, в моем мире демоном был бы ты, но в текущий момент я в твоем мире, поэтому демон я.
Роберт Асприн. Другой отличный миф



Демонами в мире Unix традиционно называются процессы, которые не взаимодействуют с пользователем напрямую. У процесса-демона нет управляющего терминала и, соответственно, нет пользовательского интерфейса. Для управления демонами приходится использовать другие программы. Само название «демоны» возникло благодаря тому, что многие процессы этого типа большую часть времени проводят в ожидании какого-то события. Когда это событие наступает, демон активизируется (выпрыгивает, как чертик из табакерки), выполняет свою работу и снова засыпает в ожидании события. Следует отметить, что многие демоны, такие как, например, web-сервер или сервер баз данных, могут отбирать на себя практически все процессорное время и другие ресурсы системы. Такие демоны гораздо больше работают, чем спят.

Скажу честно, что вам, уважаемый читатель, вряд ли придется заниматься созданием собственного демона, поскольку круг задач, для которых он может понадобиться, не так уж и велик. Область применения демонов – создание таких приложений, которые могут (и должны) выполняться без участия пользователя. Обычно, это разного рода серверы. Тем не менее, демоны задействуют многие важные элементы системы, и знание принципов их работы способствует пониманию принципов Unix/Linux в целом. Мы рассмотрим работу демонов на примере простого (очень простого) сетевого сервера *aahzd* (надеюсь, что поклонники творчества Асприна меня поймут и простят), способного отвечать на запросы клиентов. Исходный код нашего сервера (вы найдете его на диске в файле **aahzd.c**) представляет собой доработанный вариант открытого демонстрационного (простите за невольный каламбур) демона, написанного Давидом Жилье [David Gillies]. Те места исходного текста, в которых я внес изменения, помечены в файле **aahzd.c** комментарием "Note by A.B. ..." Поскольку разговор о демонах вызывает у меня трепет, я не буду долго теоретизировать и сразу перейду к коду (функции main() нашего демона)

```
volatile sig_atomic_t    gGracefulShutdown=0;
volatile sig_atomic_t    gCaughtHupSignal=0;
int             gLockFileDesc=-1;
int             gMasterSocket=-1;
const int       gaahzdPort=30333;
const char *const    gLockFilePath = "/var/run/aahzd.pid";

int main(int argc,char *argv[])
{
   int                result;
   pid_t              daemonPID;
   if (argc > 1)
     {
       int fd, len;
        pid_t pid;
       char pid_buf[16];

       if ((fd = open(gLockFilePath, O_RDONLY)) < 0)
       {
          perror("Lock file not found. May be the server is not running?");
          exit(fd);
       }
       len = read(fd, pid_buf, 16);
       pid_buf[len] = 0;
       pid = atoi(pid_buf);
       if(!strcmp(argv[1], "stop"))
       {
          kill(pid, SIGUSR1);
          exit(EXIT_SUCCESS);
       }
       if(!strcmp(argv[1], "restart"))
       {
          kill(pid, SIGHUP);
          exit(EXIT_SUCCESS);
       }
       printf ("usage %s [stop|restart]\n", argv[0]);
       exit (EXIT_FAILURE);
     }
   if((result = BecomeDaemonProcess(gLockFilePath, "aahzd",
LOG_DEBUG, &gLockFileDesc, &daemonPID))<0)
```

» **Месяц назад** Мы научились синхронизировать потоки POSIX.

```
    {
    perror("Failed to become daemon process");
    exit(result);
    }
  if((result = ConfigureSignalHandlers())<0)
    {
    syslog(LOG_LOCAL0|LOG_INFO, "ConfigureSignalHandlers failed, 
errno=%d", errno);
    unlink(gLockFilePath);
    exit(result);
    }
  if((result = BindPassiveSocket(INADDR_ANY, gaahzdPort, 
&gMasterSocket))<0)
    {
    syslog(LOG_LOCAL0|LOG_INFO, "BindPassiveSocket failed, errno=%d", 
errno);
    unlink(gLockFilePath);
    exit(result);
    }

  do
    {
    if(AcceptConnections(gMasterSocket)<0)
      {
      syslog(LOG_LOCAL0|LOG_INFO,"AcceptConnections failed, 
errno=%d",errno);
      unlink(gLockFilePath);
      exit(result);
      }

    if((gGracefulShutdown==1)&&(gCaughtHupSignal==0))
      break;

    gGracefulShutdown=gCaughtHupSignal=0;
    }while(1);
  TidyUp();
  return 0;
}
```

Сейчас мы пропустим блок операторов if (argc > 1) {...} (мы вернемся к нему позже) и рассмотрим основные этапы работы демона. Функция BecomeDaemonProcess() превращает обычный консольный процесс Linux в процесс-демон. Функция ConfigureSignalHandlers() настраивает обработчики сигналов процесса-демона, а функция BindPassiveSocket() открывает сокет TCP/IP для прослушивания входящих запросов. Далее следует цикл, в котором сервер обрабатывает запросы. Многие сетевые серверы, получив запрос, создают дочерний процесс для его обработки. Так достигается возможность параллельной обработки запросов. Другие серверы используют для этого потоки. Скажем сразу, из соображений простоты наш сервер обрабатывает запросы в последовательном (блокирующем) режиме. Мы ведь не ожидаем, что наш демонстрационный сервер будет получать много запросов, не так ли?

Нормальный выход из цикла обработки запросов происходит при получении процессом сигнала SIGUSER1. После выхода из цикла процесс вызывает функцию TidyUp() и завершает работу. Мы, безусловно, можем завершить процесс-демон, послав ему сигнал SIGKILL (SIGTERM и некоторые другие), но пользовательский сигнал SIGUSER1 гарантирует вежливое завершение процесса. В нашем случае «вежливое завершение» означает, что сервер ответит на текущий запрос перед тем, как завершиться, и удалит свой pid-файл.

Рассмотрим теперь подробнее функцию BecomeDaemonProcess(), благодаря которой обычный процесс Linux становится демоном. Мы не будем рассматривать тексты функций целиком, так как иначе статья будет состоять исключительно из листингов (на самом деле, длина листингов даже превышает пространство, выделенное под статью).

Рассмотрим для начала фрагмент:

```
chdir("/");
```

Мы делаем корень файловой системы текущим каталогом для процесса-демона. Будучи запущен, наш демон может работать вплоть до перезагрузки системы. Поэтому текущей должна быть выбрана такая файловая система, которая не может быть размонтирована. Далее следует вызов

```
lockFD = open(lockFileName, O_RDWR|O_CREAT|O_EXCL, 0644);
```

Каждый процесс-демон создает так называемый pid-файл (или файл блокировки). Этот файл обычно содержится в директории **/var/run** и имеет имя **daemon.pid**, где "daemon" – имя демона. Файл блокировки содержит значение PID процесса демона. Этот файл важен по двум причинам. Во-первых, его наличие свидетельствует о том, что в системе уже запущен процесс-демон. Дело в том, что большинство демонов, включая наш, следят за тем, чтобы в системе был запущен только один экземпляр процесса (это логично, если учесть, что демоны часто обращаются к неразделяемым ресурсам, таким, как сетевые порты). Завершаясь, процесс-демон удаляет pid-файл, указывая тем самым, что можно запустить другой экземпляр процесса. Однако работа демона не всегда завершается нормально, и тогда на диске остается pid-файл несуществующего процесса. Это, казалось бы, может стать непреодолимым препятствием к повторному запуску демона, но на самом деле, демоны успешно справляются с такими ситуациями. Обнаружив на диске pid-файл, демон считывает из него значение PID и с помощью функции kill(2) проверяет, существует ли в системе процесс с указанным PID. Если процесс существует, значит, пользователь пытается запустить демона повторно. В этом случае программа выводит соответствующее сообщение и завершается. Если процесса с указанным PID в системе нет, значит, pid-файл принадлежал аварийно завершенному процессу. Здесь программа обычно советует пользователю удалить pid-файл (ответственность в таких делах всегда лучше переложить на пользователя) и попытаться запустить ее еще раз. Может, конечно, случиться и так, что после аварийного завершения демона на диске останется его pid-файл, а затем какой-то другой процесс получит соответствующий pid. В этой ситуации для демона все будет выглядеть так, как будто его копия уже работает в системе, и запустить демон повторно вы не сможете. К счастью, описанная ситуация крайне маловероятна.

Далее демон вызывает функцию fork(3), которая создает копию его процесса. Родительский процесс при этом завершается:

```
curPID=fork();
  switch(curPID)
    {
    case 0: /* we are the child process */
     break;
    case -1: /* error - bail out (fork failing is very bad) */
     fprintf(stderr,"Error: initial fork failed: %s\n",
          strerror(errno));
     return -1;
     break;
    default: /* we are the parent, so exit */
     exit(0);
     break;
    }
```

Делается это для того, чтобы процесс перестал быть лидером сессии. Сессиями в Unix называются наборы групп процессов. Каждая сессия связана с одним управляющим терминалом, на который осуществляется вывод, и с которого выполняется ввод данных. Только одна из групп процессов, входящих в сессию, имеет доступ к терминалу. Эта группа именуется foreground (приоритетной). В каждой сессии есть процесс-родоначальник, который называется лидером сессии.

Если процесс-демон запущен с терминала, он должен, прежде всего, отключиться от этого терминала. Самый простой способ сделать это – начать новую сессию, так как по умолчанию сессия не связана ни с каким терминалом. Чтобы процесс мог начать новую сессию, он сам не должен быть лидером сессии. Полученный в результате вызова fork() дочерний процесс таковым точно не является. Он смело начинает новую сессию с помощью вызова функции setsid(2). При этом наш процесс становится лидером (и единственным участником) новой сессии.

```
if(setsid()<0)
return -1;
```



Далее некоторые руководства рекомендуют снова вызвать fork(), чтобы новый процесс перестал быть лидером новой сессии (в System V лидер сессии может автоматически получить управляющий терминал при некоторых условиях). В Linux повторный вызов fork() мы делать не будем.

Стоит отметить, что теперь у нашего процесса новый PID, который мы снова должны записать в pid-файл демона. Мы записываем значение PID в файл в строковом виде (а не как переменную типа pid_t). Делается это для удобства пользователя, чтобы значение PID из pid-файла можно было считать с помощью cat. Например:

```
kill `cat /var/run/aahz.pid`
```

Далее мы закрываем все файловые дескрипторы, которые мы могли унаследовать от родительских процессов.

```
numFiles = sysconf(_SC_OPEN_MAX);
for(I = numFiles-1;I >= 0; --i)
{
if(i!=lockFD)
close(i);
}
```

Функция sysconf() с параметром _SC_OPEN_MAX возвращает максимально возможное количество дескрипторов, которые может открыть наша программа. Мы вызываем функцию close() для каждого дескриптора (независимо от того, открыт он или нет), за исключением дескриптора pid-файла, который должен оставаться открытым.

Дескрипторы стандартных потоков ввода, вывода и ошибок не следует оставлять закрытыми, так как многие функции стандартной библиотеки предполагают, что они открыты. Мы открываем эти дескрипторы и перенаправляем их на **/dev/null**:

```
stdioFD = open("/dev/null", O_RDWR);
dup(stdioFD);
dup(stdioFD);
```

Теперь мы можем быть уверены, что демон не получит доступа к какому-либо терминалу. Тем не менее, у демона должна быть возможность выводить куда-то сообщения о своей работе. Традиционно для этого используются файлы журналов (log-файлы). Файлы журналов для демона подобны черным ящикам самолетов. Если в работе демона произошел какой-то сбой, пользователь может проанализировать файл журнала и (при определенном везении) установить причину сбоя. Ничто не мешает нашему демону открыть свой файл журнала, но это не очень удобно. Большинство демонов пользуются услугами утилиты *syslog*, ведущей журналы множества системных событий. Доступ к журналу *syslog* можно открыть с помощью функции openlog(3):

```
openlog(logPrefix, LOG_PID|LOG_CONS|LOG_NDELAY|LOG_NOWAIT, LOG_
LOCAL0);
(void)setlogmask(LOG_UPTO(logLevel));
```

Первый параметр функции openlog() – префикс, который будет добавляться к каждой записи в системном журнале. Далее следуют различные опции *syslog*. Функция setlogmask(3) позволяет установить уровень приоритета сообщений, которые записываются в журнал событий. При вызове функции BecomeDaemonProcess() мы передаем в параметре logLevel значение LOG_DEBUG. В сочетании с макросом LOG_UPTO это означает, что в журнал будут записываться все сообщения с приоритетом, начиная с наивысшего и заканчивая LOG_DEBUG. Наконец, вызов

```
setpgrp();
```

создает новую группу процессов, идентификатором которой является идентификатор текущего процесса. На этом работа функции BecomeDaemonProcess() завершается, так как теперь наш процесс стал настоящим демоном.

Функция ConfigureSignalHandlers() настраивает обработчики сигналов. Сигналы, которые получит наш демон, можно разделить на три группы: игнорируемые, «фатальные» и обрабатываемые. Вызывая функцию

```
signal(SIGUSR2, SIG_IGN);
```

мы указываем, что наш демон должен игнорировать сигнал SIGUSR2. Аналогично мы поступаем с сигналами SIGPIPE, SIGALRM, SIGTSTP, SIGPROF, SIGCHLD. Сигналы SIGQUIT, SIGILL, SIGTRAP, SIGABRT, SIGIOT, SIGBUS, SIGFPE, SIGSEGV, SIGSTKFLT, SIGCONT, SIGPWR и SIGSYS относятся к категории «фатальных». Мы не можем их игнорировать, но и продолжать выполнение процесса-демона после получения одного из них нежелательно. Мы назначаем всем этим сигналам обработчик FatalSigHandler, например:

```
signal(SIGQUIT, FatalSigHandler);
```

FatalSigHandler() записывает в журнал событий информацию о полученном сигнале и завершает процесс, вызвав перед этим функции closelog() и TidyUp(), освобождающие все занятые процессом ресурсы:

```
void FatalSigHandler(int sig)
{
#ifdef _GNU_SOURCE
  syslog(LOG_LOCAL0|LOG_INFO,"caught signal: %s - exiting",strsignal(sig)
);
#else
  syslog(LOG_LOCAL0|LOG_INFO,"caught signal: %d - exiting",sig);
#endif
  closelog();
  TidyUp();
  _exit(0);
}
```

Три сигнала, относящихся к категории обрабатываемых – SIGTERM, SIGUSR1 и SIGHUP, обрабатываются по-разному:

```
sigtermSA.sa_handler = TermHandler;
sigemptyset(&sigtermSA.sa_mask);
sigtermSA.sa_flags = 0;
sigaction(SIGTERM,&sigtermSA,NULL);
sigusr1SA.sa_handler = Usr1Handler;
sigemptyset(&sigusr1SA.sa_mask);
sigusr1SA.sa_flags = 0;
sigaction(SIGUSR1,&sigusr1SA,NULL);
sighupSA.sa_handler = HupHandler;
sigemptyset(&sighupSA.sa_mask);
sighupSA.sa_flags = 0;
sigaction(SIGHUP,&sighupSA,NULL);
```

Обработчик TermHandler() вызывает функцию TidyUp() и завершает процесс. Обработчик Usr1Handler() делает в системном журнале запись о вежливом завершении процесса и присваивает переменной gGracefulShutdown значение **1** (что, как вы помните, приводит к выходу из цикла обработки запросов, когда цикл будет готов к этому). Обработчик сигнала HupHandler() также делает запись в системном журнале, после чего присваивает значение **1** переменным gGracefulShutdown и gCaughtHupSignal. В реальной жизни получение сигнала SIGHUP приводит к перезапуску демона, сопровождаемому повторным прочтением файла конфигурации (первый раз демон читает этот файл во время запуска) и переустановкой значений записанных в нем параметров. Именно необходимость прочесть повторно файл конфигурации является наиболее частой причиной перезапуска демонов. У нашего демона файла конфигурации нет, так что в процессе перезапуска делать ему особенно нечего.

Функция BindPassiveSocket() открывает для прослушивания порт сервера (в нашем случае это порт 30333) на всех доступных сетевых интерфейсах и возвращает соответствующий сокет:

```
int BindPassiveSocket(const int portNum,
                              int *const boundSocket)
{
struct sockaddr_in sin;
int          newsock, optval;
size_t optlen;
memset(&sin.sin_zero, 0, 8);
sin.sin_port = htons(portNum);
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = htonl(INADDR_ANY);
if((newsock= socket(PF_INET, SOCK_STREAM, 0))<0)
return -1;
optval = 1;
optlen = sizeof(int);
setsockopt(newsock, SOL_SOCKET, SO_REUSEADDR,&optval,optlen);
if(bind(newsock,(struct sockaddr*)&sin,sizeof(struct sockaddr_in))<0)
```

```
return -1;
if(listen(newsock,SOMAXCONN)<0)
return -1;
*boundSocket = newsock;
return 0;
}
```

Тем, кто читал статью этой серии, посвященную сокетам (см. LXF83), должно быть понятно, что здесь происходит. Отметим только одну интересную деталь. Если предыдущий, уже закрытый, сокет, связанный с данным портом, находится в состоянии TIME_WAIT, между закрытием старого и открытием нового сокета может произойти задержка, равная двум периодам жизни сегмента (задержка может составлять до двух минут). Для того, чтобы при повторном запуске демона нам не пришлось ждать, мы используем функцию setsockopt() с параметром SO_REUSEADDR.

Функция AcceptConnections() обрабатывает запросы последовательно, используя блокирующий вызов accept():

```
int AcceptConnections(const int master)
{
   int     proceed = 1, slave, retval = 0;
   struct sockaddr_in      client;
   socklen_t      clilen;
   while((proceed==1)&&(gGracefulShutdown==0))
   {
      clilen = sizeof(client);
      slave = accept(master,(struct sockaddr *)&client,&clilen);
      if(slave<0) /* accept() failed */
      {
         if(errno == EINTR)
            continue;
         syslog(LOG_LOCAL0|LOG_INFO,"accept() failed: %m\n");
         proceed = 0;
         retval = -1;
      }
      else
      {
         retval = HandleConnection(slave); /* process connection */
         if(retval)
            proceed = 0;
      }
      close(slave);
   }
   return retval;
}
```

Это не лучший образ поведения демона, но если мы начнем описывать параллельную обработку запросов, редакция не выдержит. Переменная proceed, совместно с переменной gGracefulShutdown, указывает, должна ли программа продолжать обрабатывать запросы. Если очередной вызов connect() или HandleConnection() вернул сообщение об ошибке, этой переменной присваивается **0** и обработка запросов прекращается. Новый сокет, полученный в результате вызовы accept(), передается функции HandleConnection().

```
int HandleConnection(const int slave)
{
char       readbuf[1025];
size_t     bytesRead;
const size_t               buflen=1024;
int            retval;
retval = ReadLine(slave, readbuf, buflen, &bytesRead);
if(retval==0)
           WriteToSocket(slave, readbuf, bytesRead);
return retval;
}
```

Функция HandleConnection() считывает переданную клиентом строку и тут же возвращает ее клиенту. Затем функция AcceptConnections() закрывает соединение, открытое в результате вызова accept(). Функции ReadLine() и WriteToSocket() тривиальны, и рассматривать их мы не будем. Если где-то в цепочке вызовов AcceptConnections(), HandleConnection(), ReadLine() и WriteToSocket() возникла ошибка, информация об ошибке будет передаваться вверх по цепочке до тех пор, пока не достигнет функции main(), приводя к немедленному завершению работы демона с соответствующей записью в журнал системных сообщений.

Рассмотрим, наконец, функцию TidyUp(), к которой обращаются многие функции сервера перед тем, как завершить его работу.

```
void TidyUp(void)
{
   if(gLockFileDesc!=-1)
      {
      close(gLockFileDesc);
      unlink(gLockFilePath);
      gLockFileDesc=-1;
      }

   if(gMasterSocket!=-1)
      {
      close(gMasterSocket);
      gMasterSocket=-1;
      }
}
```

Задача TidyUp() – «прибрать мусор» за демоном. В принципе, без этой функции можно обойтись, так как после завершения процесса система сама закроет все его дескрипторы, но правила хорошего тона требуют явного высвобождения всех ресурсов, выделенных явным образом.

Если вы скомпилируете программу-демон с помощью команды

```
gcc aahz.c -o aahz
```

то сможете запустить демона командой

```
# ./aahz
```

Поскольку демон нуждается в доступе к директории **/var/run**, запускать программу нужно от имени root. Сразу после запуска вы снова увидите приглашение командной строки, что для демонов совершенно нормально. Если бы сервер *aahzd* выполнял что-нибудь полезное, команду его запуска можно было бы прописать в одном из сценариев запуска системы в директории **/etc/init.d**, но мы этого делать не будем. После того как сервер запущен, вы можете дать команду

```
telnet 127.0.0.1 30333
```

Будет установлено соединение с сервером, который продублирует строку, введенную вами в консоли *telnet*, и закроет соединение.

Вернемся теперь к начальным строкам функции main(). Хотя мы можем получить PID демона из его pid-файла и управлять демоном с помощью команды *kill*, такой вариант многими считается неудобным. Часто для управления используется сам исполняемый файл демона, запускаемый со специальными аргументами командной строки. Наш демон понимает две команды – *stop* (завершение работы демона) и *restart* (перезапуск). Посмотрим, как поведет себя демон, запущенный с аргументами командной строки. В этом случае в начале программы демон считывает значение pid из собственного файла. Если открыть pid-файл не удается, значит, скорее всего, демон не запущен, и управляющему режиму просто нечего делать. Если значение pid получено, процесс, управляющий демоном, с помощью функции kill() посылает демону соответствующий сигнал.

Демоны не рассчитаны на то, чтобы получать какую-либо информацию от пользователя. Собственную информацию они передают другим программам либо записывают в журналы системных событий. В следующей статье мы сосредоточимся на программах, которые ведут себя совершенно иначе, и рассмотрим консольный ввод-вывод. LXF

Через месяц Мы займемся интерактивными утилитами и консольным вводом/выводом.

Адресная

ЧАСТЬ 1: Театр, как известно, начинается с вешалки, а приложения уровня предприятий – с адресной книги: надо же где-то хранить сведения о клиентах. **Александр Бабаев** готов познакомить вас с азбукой Java EE.



Наш эксперт

Александр Бабаев
Разработчик открытой мультиблог-системы jDnevnik – победитель конкурсов IBM WAS CE Contest 2006 и конкурса проектов для разработчиков на Java - Java конкурс 2005.

Один из самых распространенных языков программирования для Интернета на настоящий момент – это PHP. Этот язык изначально задумывался для создания домашних страничек, небольших сайтов – и там работает замечательно.

В отличие от PHP, Java (та ее часть, которая помогает разрабатывать серверные приложения) сразу создавалась для крупных приложений уровня предприятия (так называемых Enterprise Applications). Поэтому в Java есть множество механизмов, которые помогают быстро строить очень крупные приложения. Очень крупные – это десятки тысяч транзакций в минуту. Сложных транзакций.

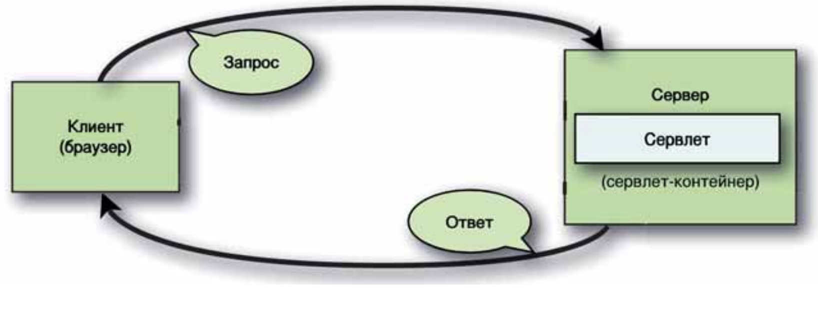
Обо всех возможностях этой технологии не рассказать ни в рамках одной статьи, ни даже в десятке. Но основные блоки JEE (Java Enterprise Edition), которые полезны не только в крупных приложениях, но и в средних и небольших, мы изучим обязательно. А на основании этого опыта можно будет двигаться дальше, читать книги, разрабатывать сложные системы.

Сервлеты

Для начала решим, что же такое сервлет (servlet)? Этим термином принято называть серверное приложение, но чем отличается серверное приложение от клиентского? Клиентское ПО формирует запросы и отсылает их серверному, а задача серверной части – обработать запрос и вернуть на него ответ (**рис. 1**).

На стороне клиента присутствуют только действия, которые выполняются браузером (показ формы на HTML-странице, формирование стандартизованного запроса).

Серверная сторона в Java обычно состоит из контейнера, который содержит один или несколько сервлетов. Контейнер получает запрос, решает, какому сервлету он предназначен, и запускает на выполнение этот сервлет. Формально происходит примерно следующее:



› **Рис. 1. Процесс обработки запроса**

- Пришедший от клиента запрос анализируется сервлет-контейнером, который создает на его основе специальный объект javax.servlet.ServletRequest.
- Контейнер решает (как – обычно описывается в специальном конфигурационном файле), какому сервлету предназначен запрос, и передает ему созданный на предыдущем шаге объект javax.servlet.ServletRequest, а также объект javax.servlet.ServletResponse – шаблон ответа, в который сервлет пишет все, что нужно возвратить клиенту.
- Сервлет обрабатывает запрос, заполняет объект javax.servlet.ServletResponse и после этого завершает работу.
- Контейнер получает заполненный объект javax.servlet.ServletResponse, после чего преобразует его в стандартизованное сообщение и передает клиенту.

Что нам потребуется?

Мы предполагаем, что читатель знаком с тем, что такое HTML, и пробовал делать «странички на PHP» или чем-то аналогичном. Если слово-

Коротко о HTTP

Протокол HTTP был предложен Тимом Бернерсом-Ли, работавшим тогда в CERN, в марте 1990 года как механизм для доступа к документам в Интернете и облегчения навигации посредством использования гипертекста. В HTTP разделяются понятия «запрос» и «ответ», то есть то, как браузер запрашивает страницу и то, чем сервер на это отвечает. И запрос, и ответ состоят из заголовка и тела. В запросе присутствует еще и строка запроса. Заголовок описывает передаваемые данные, внутри тела передается обычно результат (страница HTML или файл). Существует несколько типов запросов, но наиболее распространены два:

- **GET** Запрашивает содержимое указанного ресурса.
- **POST** Передает пользовательские данные (например, из формы HTML) заданному ресурсу. Данные включаются в тело запроса.

Вот пример (не точный) простого HTTP-диалога:

Запрос

```
GET /site/index.html HTTP/1.1
Host: ru.wikipedia.org
```

Ответ

```
HTTP/1.0 200 OK
Server: Apache
Content-Type: text/html; charset=utf-8
Content-Length: 2121
(Дальше идет 2121 байт текста странички index.html)
```

книга

сочетания «GET-запрос», «POST-запрос» являются для вас китайской грамотой – прочтите врезку «Коротко об HTTP» (и запаситесь терпением). Подробнее про протокол HTTP можно прочитать в RFC2616, например тут: http://www.w3.org/Protocols/rfc2616/rfc2616.html.

Для работы нам понадобятся следующие инструменты:

» JDK 5 (или JDK 6) – если вы выполняли все задания предыдущей серии уроков Java, нужный дистрибутив, скорее всего, уже имеется в вашей системе (вы ведь выполняли задания, не так ли?). JDK можно бесплатно загрузить с сайта http://java.sun.com или установить из репозиториев вашего дистрибутива Linux. После установки JDK необходимо убедиться, что пути к каталогу **$JDK/bin** прописаны в переменной окружения PATH, а каталоги, в которых располагаются библиотеки классов, перечислены в CLASSPATH.

» Ваш любимый текстовый редактор.

» Библиотека Jetty (http://www.mortbay.org/). В принципе, это полноценный web-сервер, который может очень и очень многое. Нам будет полезна его особенность, позволяющая встроить web-сервер в наше приложение.

Все необходимые библиотеки есть на прилагаемом компакт-диске.

Что мы будем делать?

Для примера мы возьмем и напишем корпоративную адресную книгу, в которой будем сохранять телефоны сотрудников, причем каждый сможет добавить телефоны нужных людей, и все смогут просмотреть сохраненные телефоны. Итак, начнем.

Создадим иерархию каталогов для проекта, например, такую:

» **~/Programming/AddressBook/**.

» **build** – каталог для скомпилированных классов.

» **src** – исходные тексты.

» **libs** – каталог для библиотек.

Заполним каталоги. Для начала возьмите Jetty, распакуйте архив, найдите в нем файлы **jetty-6.1.0rc3.jar**; **jetty-util-6.1.0rc3.jar**; **servlet-api-2.5.jar** и поместите их в каталог **libs** (можно взять и другую версию, правда, с известной долей осторожности – иногда меняются API, тогда наша программа просто не скомпилируется).

Теперь настало время придумать, как будет работать наша телефонная книга. Разумно будет создать четыре основных страницы:

» Главная – на ней будут ссылки помещены на остальные страницы приложения.

» Страница добавления нового контакта.

» Страница редактирования контакта.

» Страница поиска/просмотра контактов.

Классов в нашем приложении будет всего три:

AddressBook. Главный класс, который запускается и «висит» в памяти, обрабатывая запросы пользователей. Обычно эту роль выполняет уже упомянутый сервлет-контейнер (роль которого может выполнять, например, Apache Tomcat), но мы избавились от него, использовав Jetty.

AddressBookHandler. Класс-обработчик запросов пользователей. Именно этот класс и называется сервлетом. Он обычно подключается к сервлет-контейнеру через специфический конфигурационный файл, но мы избавились и от этого – спасибо, Jetty!

Contact. Класс, который содержит поля контакта. Эти поля будут выводиться на странице и редактироваться. Его код очень простой:

Листинг 1. Контакт

```
import java.io.*;
public class Contact implements Serializable {
  private String _name = "";
  private String _number = "";
  private String _comment = "";

  public String getName() { return _name; }
  public String getNumber() { return _number; }
  public String getComment() { return _comment; }

  public void setData(String aName, String aNumber, String aComment) {
    _name = aName;
    _number = aNumber;
    _comment = aComment;
  }
}
```

Основной класс/класс запуска приложения (AddressBook)

Этот класс содержит метод main, инициализирует Jetty и подключает сервлет (AddressBookHandler, **листинг 2**).

Листинг 2. Класс запуска приложения

```
import org.mortbay.jetty.*;
import org.mortbay.jetty.nio.*;

import java.util.*;
import java.io.*;

public class AddressBook {
  private void start() throws Exception {
    try {
      _contactsByName = (SortedMap<String, Contact>) new
XStream().
          fromXML(new FileReader("contacts.xml"));
    } catch (Exception e) { }

    Server _jettyServer = new Server();
    Connector connector = new SelectChannelConnector();
    connector.setPort(8081);
    _jettyServer.setConnectors(new Connector[]{connector});
    _jettyServer.setStopAtShutdown(true);
    Handler handler = new AddressBookHandler(this);
    _jettyServer.setHandler(handler);
    _jettyServer.start();
  }
  public static void main(String[] args) throws Exception {
    new AddressBook().start();
```

```
    }
}
```

Также этот класс содержит саму адресную книгу:

Листинг 3. Адресная книга

```
private SortedMap<String, Contact> _contactsByName = new TreeMap<St
ring,Contact>();

public SortedMap<String, Contact> getContactsByName() {
    return _contactsByName;
}

public Contact getContactByNumber(String aNumber) {
    for (Map.Entry<String, Contact> entry : _contactsByName.entrySet()) {
        if (entry.getValue().getNumber().equals(aNumber)) {
            return entry.getValue();
        }
    }
    return null;
}
```

и методы, которые позволяют добавлять/удалять/редактировать контакты:

Листинг 4. Работа с контактами

```
public void addContact(String aName, String aNumber, String aComment)
throws IOException {
    Contact contact = new Contact();
    contact.setData(aName, aNumber, aComment);
    _contactsByName.put(aName, contact);
    saveContactsInformation();
}

public void removeContactByNumber(String aNumber) throws
IOException {
    Contact contact = getContactByNumber(aNumber);
    if (contact != null) {
        _contactsByName.remove(contact.getName());
        saveContactsInformation();
    }
}

public void editContact(String aEditNumber, String aName, String
aNumber, String aComment)
throws IOException {
    Contact contact = getContactByNumber(aEditNumber);
    if (contact != null) {
        contact.setData(aName, aNumber, aComment);
        saveContactsInformation();
    }
}

private void saveContactsInformation() throws IOException {
    new XStream().toXML(_contactsByName, new FileWriter(“contacts.
xml”));
}
```

Класс-обработчик (AddressBookHandler)

Обработка событий пользователя концентрируется в одном методе (public void handle(…)) класса AddressBookHandler. Самые главные параметры этого метода – второй aHttpServletRequest и третий aHttpServletResponse. Request – это запрос. В нем хранятся все данные, которые передал нам браузер пользователя. А Response – это объект, в который нужно записать все, что требуется передать пользователю. Например, на любой запрос пользователя можно отвечать страницей с гордым заголовком «адресная книга» (**листинг 5**):

Листинг 5. Код обработчика

```
import org.mortbay.jetty.handler.*;
import org.mortbay.jetty.*;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;

public class AddressBookHandler extends AbstractHandler {
    private AddressBook _addressBook = null;

    public AddressBookHandler(AddressBook aAddressBook) {
        _addressBook = aAddressBook;
    }

    public void handle(String aTarget,            HttpServletRequest
aRequest,
            HttpServletResponse aResponse,
            int aDispatchMode)
            throws IOException, ServletException {

        ((Request) aRequest).setHandled(true);
        aRequest.setCharacterEncoding(“utf-8”);
        aResponse.setCharacterEncoding(“utf-8”);
        aResponse.addHeader(“Content-type”, “text/html; charset=utf-8”);
        PrintWriter writer = aResponse.getWriter();
        writer.write(“<html><head>” +
            “<title>Адресная книга</title>” +
            “</head><body><h1>Адресная книга</h1>”);
    }
}
```

Откомпилируйте полученный код:

```
cd ~/Programming/AddressBook/src
javac –encoding utf-8 -cp ../libs/jetty-6.1.0rc3.jar:../libs/jetty-util-6.1.0rc3.
jar:../libs/servlet-api-2.5.jar -d ../build *.java
```

И запустите (не забыв подключить все требуемые библиотеки):

```
cd ~/Programming/AddressBook/build
java -cp ../libs/jetty-6.1.0rc3.jar:../libs/jetty-util-6.1.0rc3.jar:../libs/servlet-
api-2.5.jar:. AddressBook
```

Терминал выводит несколько строчек и «зависает». Все правильно, сервер запустился. Запускать наше приложение будем пока именно так, чтобы его можно было легко выключить (Ctrl+C). Теперь откройте браузер и наберите в адресной строке: **http://localhost:8081**. Вот он, наш заголовок!

Вернемся к обработчику и попробуем уяснить, как различать запросы пользователей. Для этого в обработчике есть две возможности. Самая простая – посмотреть на первый параметр (aTarget), который содержит «файл» запроса (например в URL'е **«http://www.linuxformat.ru/index.html»** aTarget будет **«/index.html»**); посложнее – анализировать сам объект запроса. Воспользуемся первым вариантом, дописав в конец метода handle (прямо перед закрывающей скобкой) следующее:

Листинг 6. Более сложный обработчик с разбором адреса

```
if (“/”.equals(aTarget)) {
    writer.write(“<a href=\”/add\”>Добавить запись</a><br/>”);
    writer.write(“<a href=\”/view\”>Просмотреть записи</a><br/>”);
} else {
    writer.write(“<a href=\”/\”>На главную</a><br/>”);

    if (“/add”.equals(aTarget)) {
        writer.write(“<h2>Добавление записи</h2>”);
    } else if (“/view”.equals(aTarget)) {
        writer.write(“<h2>Просмотр записей</h2>”);
    } else if (“/edit”.equals(aTarget)) {
```

```
        writer.write("<h2>Редактирование записи</h2>");
    } else if ("/remove".equals(aTarget)) {
        writer.write("<h2>Удаление записи</h2>");
    }
}

writer.write("</body></html>");
```

Попробуйте снова откомпилировать и запустить приложение, предварительно остановив предыдущую версию сервера. Обновите страницу в браузере – и побегайте по только что созданным страницам нашего сайта (корневая, она же **«/»**, **«/add»**, **«/edit»**, **«/remove»**).

Теперь в соответствующих ветках if можно написать обработчики страничек. Для упрощения вынесем эти обработчики в отдельные методы (**листинг 7**):

Листинг 7. Добавляем обработчики отдельных действий

```
        if ("/add".equals(aTarget)) {
            writer.write("<h2>Добавление записи</h2>");
            handleAdd(aRequest, writer);
        } else if ("/view".equals(aTarget)) {
            writer.write("<h2>Просмотр записей</h2>");
            handleView(writer);
        } else if ("/edit".equals(aTarget)) {
            writer.write("<h2>Редактирование записи</h2>");
            handleEdit(aRequest, writer);
        } else if ("/remove".equals(aTarget)) {
            writer.write("<h2>Удаление записи</h2>");
            handleRemove(aRequest, writer);
        }
```

Осталось понять, как именно обрабатывать соответствующие страницы. Рассмотрим, для примера, метод handleAdd() – остальные можно написать по аналогии:

Листинг 8. Обработчик добавления записи

```
    private void handleAdd(HttpServletRequest aRequest, PrintWriter
aWriter)
            throws IOException {
        if (aRequest.getParameter("name") != null) {
            _addressBook.addContact(aRequest.getParameter("name"),
                    aRequest.getParameter("number"), aRequest.getParameter(
"comment"));
            outputMessage(aWriter, "Контакт добавлен");
        }

        outputForm(aWriter, aRequest, "", "", "");
    }
```

В приведенном обработчике есть вызов метода aRequest.getParameter("name"). Этот метод выдает значение параметра, который передает браузер из формы (<input type="text" name="name"/>). Причем сервлету все равно, каким методом (GET/POST) был передан параметр. Конкретно для обработки добавления контакта в коде проверяется, равен ли результат вызова **null**. Если так – значит, форма «не выполнялась», и добавлять нечего. В противном случае мы считываем три параметра (имя, телефон и комментарий), сохраняем контакт и выводим форму (вдруг пользователь захочет добавить еще один контакт?).

Методы outputForm и outputMessage (**листинг 9**) выводят форму (либо пустую, либо заполненную данными, которые нужно редактировать) и сообщение (чтобы пользователь понимал, что действие произошло, и как оно завершилось):

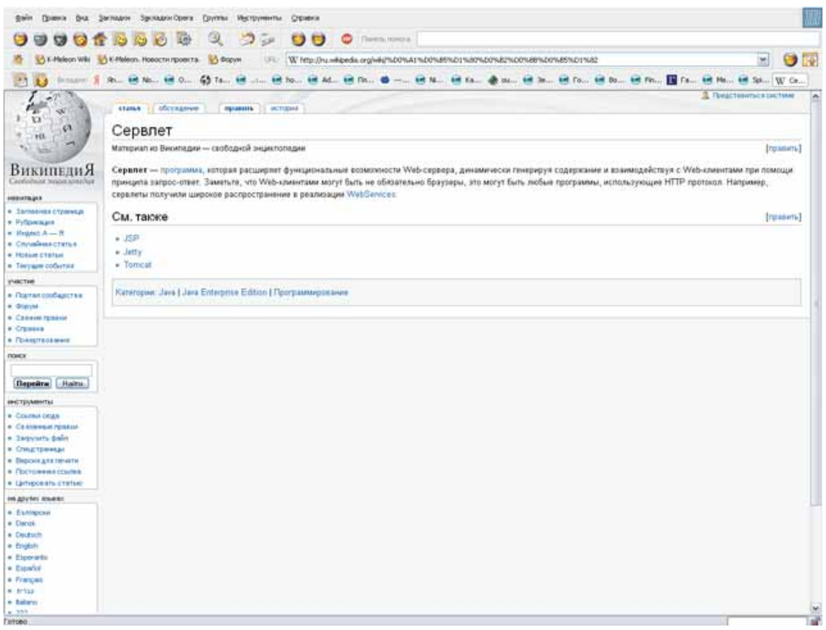
Листинг 9. Обработчик добавления записи

```
private void outputForm(PrintWriter aWriter, HttpServletRequest aRequest,
                        String aName, String aNumber, String aComment) {
    aWriter.write(
            "<form action=\"/edit\" method=\"post\">" +
            "<input type=\"hidden\" name=\"edited\" value=\"" +
                aRequest.getParameter("number") + "\"/>" +
            "<table>" +
            "<tr><td>Имя: </td><td><input type=\"text\" name=\"name\"
value=\"" +
                aName + "\"/></td></tr>" +
            "<tr><td>Телефон: </td><td><input type=\"text\" name=\"number\"
value=\"" +
                aNumber + "\"/></td></tr>" +
            "<tr><td>Примечания: </td><td><input type=\"text\" name=\
"comment\" value=\"" +
                aComment + "\"/></td></tr>" +
            "<tr><td colspan=\"2\" align=\"center\">" +
                "<input type=\"submit\" name=\"Отправить\"/></td></tr>" +
            "</table></form>");
    }

    private void outputMessage(PrintWriter aWriter, String aMessage) {
        aWriter.write("<span style=\"color: green;\">" + aMessage + "</
span>");
    }
```

Для того, чтобы проверить работу нового обработчика, закомментируйте вызовы методов, которые еще не написаны (или напишите «заглушки» в виде пустых методов). Теперь можно снова остановить сервер, скомпилировать программу и запустить ее на выполнение. Если все набрано правильно, ошибок не последует, и можно будет просматривать табличку с контактами, добавлять их, редактировать и удалять. Наша адресная книга готова к работе!

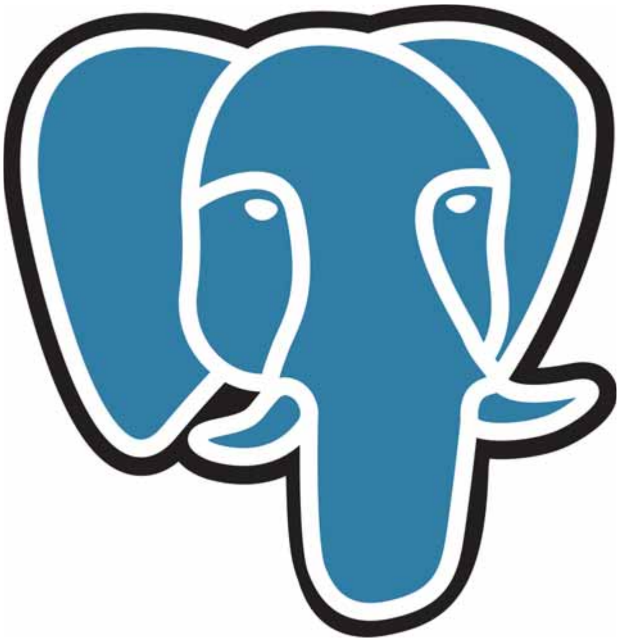
На этом уроке вы создали свое первое JEE-приложение. Конечно, за кадром осталось множество возможностей – начиная с использования сессий, контроля корректности параметров, и заканчивая отделением дизайна от логики работы сервлета – но наша программа уже вполне работоспособна. И, конечно, история не заканчивается. В следующих статьях будет продолжение, посвященное развитию приложения в соответствии с усложнением запросов компании к адресной книге. LXF



Через месяц Мы отделим логику от дизайна и познакомимся с Java Server Pages.

Интерфейсы

ЧАСТЬ 4: Интерфейсы, как известно, бывают не только пользовательскими, но и программными. С первыми мы уже успели познакомиться, а сегодня **Евгений Балдин** расскажет, какие API существуют для доступа к *PostgreSQL* из различных языков программирования.



В институте им очень дорожили, так как попутно он использовался для некоторых уникальных экспериментов и как переводчик при общении со Змеем Горынычем.
(О Кощее Бессмертном)

«Понедельник начинается в субботу»

Понятно, что любую базу данных, в принципе, можно заполнить вручную; правда, некоторые придется заполнять очень долго. СУБД – это просто хранилище, а для заполнения и доступа к хранилищу необходима инфраструктура, и эту инфраструктуру надо создавать. Вот такая она – жизнь.

Родной библиотекой для доступа к *PostgreSQL* является *libpq*. Написана она на чистом С, что тоже не удивительно, так как это основной язык родной системы. Все остальные языки важны, но безусловно, вторичны.

libpq

Чтобы общаться с базой данных, много функций не требуется: одна для открытия соединения, одна для посылки запроса, одна для получения ответа и одна для закрытия соединения. В реальности, конечно, все немного сложнее, но суть остается неизменной.

К вопросу о переносимости

Библиотека *libpq* написана на чистом С, поэтому связь с *PostgreSQL* можно организовать практически везде, где можно найти *gcc*. Мне както пришлось делать это для VAX/VMS – все решилось методом тыка, даже думать почти не потребовалось. Все данные – текст, поэтому вопрос бинарной совместимости платформ попросту отсутствует.

С чего начать

Чтобы воспользоваться вызовами *libpq*, ее необходимо иметь в системе. В Debian (Sarge) для этого надо установить пакет *PostgreSQL-dev*:

```
> sudo apt-get install postgresql-dev
```

Для доступа к функциям *libpq* необходимо включить в исходный текст соответствующие заголовки:

```
#include "libpq-fe.h"
```

Скрипт *pg_config* (*man pg_config*) позволяет получить информацию о том, куда помещаются include-файлы, библиотеки и тому подобное. Для сборки можно также использовать скрипт *libpq3-config* (*man libpq3-config* – годится, естественно, только для *libpq* третьей версии), который заведомо есть в Debian (Sarge):

```
> #сборка программы
> gcc -o "бинарник" "исходник".c -I`pg_config --includedir` \
 -lpq `/usr/bin/libpq3-config`
> cat /usr/bin/libpq3-config
#!/bin/bash
echo -lssl -lcrypto -lkrb5 -lcrypt -lresolv -lnsl -lpthread
```

libpq3-config просто выводит список всех библиотек, от которых зависит *libpq*.

Открытие и закрытие соединения

Даже открывать соединение с *PostgreSQL* можно двумя способами:

```
//открыть соединение
PGconn *PQconnectdb(const char *conninfo);
//то же, но не блокируя программу
PGconn *PQconnectStart(const char *conninfo);
//проверка статуса соединения (после PQconnectStart)
PostgresPollingStatusType PQconnectPoll(PGconn *conn);
```

PQconnectdb – обычная функция, принимающая текстовую строку conninfo, содержащую параметры для соединения с сервером, и возвращающая структуру типа PGconn с информацией о созданном соединении и успешности данной операции. В дальнейшем при передаче данных эта структура будет использоваться в качестве параметра.

Передача информации о сервере в качестве строки (conninfo) позволяет не менять внешний интерфейс вызова в случае появления дополнительных параметров и легко добавлять дополнительные опции. Пример открытия соединения:

```
const char *conninfo= "dbname = test host=localhost";
PGconn *conn=PQconnectdb(conninfo);
if (PQstatus(conn) != CONNECTION_OK) {
 fprintf(stderr, "Не удалось соединиться с базой данных: %s",
       PQerrorMessage(conn));
 /*завершаем работу*/ …}
```

Параметры передаются в форме «ключевое слово»=«значение». Пары разделяются обычным пробелом. Пробелы вокруг знака равен-

Месяц назад Мы изучали возможности PostgreSQL.

ства можно опустить. Если необходимо передать значение с пробелами, то его необходимо заключить в одинарные кавычки '«составное» «значение»'. Для передачи одинарной кавычки ее необходимо экранировать с помощью обратной косой черты \'. При отсутствии какого-либо параметра в строке conninfo его значение берется из соответствующей переменной окружения, если таковая определена. Если нет, то при открытии соединения используется значение по умолчанию.

Функции открытия соединения распознают следующие параметры и переменные окружения (кое-какие особенности опущены):

» host DNS-имя узла, на котором находится сервер *PostgreSQL*. Соответствует переменной окружения PGHOST. Значение по умолчанию: localhost.

» hostaddr Числовой адрес узла, на котором находится *PostgreSQL* (альтернатива host). Соответствует переменной окружения PGHOSTADDR. Значение по умолчанию эквивалентно: localhost.

» port Номер порта, который «слушает» POSTMASTER. Соответствует переменной окружения PGPORT. Значение по умолчанию обычно **5432**.

» dbname Имя базы данных. Соответствует переменной окружения PGDATABASE. Значение по умолчанию совпадает с системной учетной записью пользователя.

» user Имя пользователя базы данных. Соответствует переменной окружения PGUSER. Значение по умолчанию совпадает с системной учетной записью пользователя.

» password Поле пароля, если для аутентификации требуется пароль. Соответствует переменной окружения PGPASSWORD. Если аутентификация требуется, а поле не определено, то для доступа используются данные файла **~/.pgpass**. Переменная окружения PGPASSFILE может указать другой файл для проведения аутентификации.

» connect_timeout Устанавливает максимальное время ожидания соединения в секундах. С сервером и сетью всякое может случиться. Соответствует переменной окружения PGCONNECT_TIMEOUT. Значение по умолчанию равно **0**, что означает бесконечное время ожидания. Не рекомендуется устанавливать значение ожидания меньше 2 секунд.

» options Опции, посылаемые непосредственно серверу, если таковые потребуются. Соответствует переменной окружения PGOPTIONS.

» sslmode Определяет порядок действий при SSL-соединении. Принимает четыре возможных значения:
- **disable** – без шифрования
- **allow** – сначала попробовать соединиться без шифрования, а в случае неудачи постараться установить защищенное соединение,
- **prefer** – сначала попробовать установить защищенное соединение, а в случае неудачи повторить соединение без шифрования,
- **require** – выполнять только защищенное соединение. Соответствует переменной окружения PGSSLMODE.

Значение по умолчанию: **prefer**.

» krbsrvname Имя Kerberos-сервиса. Используется для аутентификации с помощью Kerberos-5[1]. Это совершенно отдельная тема, выходящая за рамки данной статьи. Соответствует переменной окружения PGKRBSRVNAME.

» PGDATESTYLE Переменная окружения, позволяющая установить представление времени и даты по умолчанию. Соответствует *SQL*-команде SET datestyle TO ...

» PGTZ Переменная окружения, позволяющая установить текущий часовой пояс. Соответствует *SQL*-команде SET timezone TO ...

» PGCLIENTENCODING Переменная окружения, позволяющая установить кодировку клиента. Соответствует *SQL*-команде SET client_encoding TO ...

Существует целый класс функций, которые позволяют получить информацию о соединении. За подробностями следует обратиться к документации, но для начала полезно знать о двух из них:

```
ConnStatusType PQstatus(const PGconn *conn);
char *PQerrorMessage(const PGconn *conn);
```

PQstatus возвращает информацию о том, как прошло соединение. Интересны состояния CONNECTION_OK – все хорошо и CONNECTION_BAD – ничего не вышло. Функция PQerrorMessage позволяет получить текстовую строку с описанием последней возникшей проблемы.

Для того, чтобы разорвать соединение, используется функция:

```
void PQfinish(PGconn *conn);
```

Внимание! Соединения следует закрывать сразу же, как только в них отпадает необходимость. Число соединений, которые поддерживает POSTMASTER, ограничено – очень легко парализовать работу базы данных, просто открывая новые соединения.

SQL-запросы

Что ж, до сервера мы уже «дозвонились», теперь пора с ним «поговорить».

Выполнение запросов

Простейший способ выполнить *SQL*-запрос – это воспользоваться функцией PQexec:

```
PGresult *PQexec(PGconn *conn, const char *command);
```

В качестве параметров функции передается структура соединения conn и строка с *SQL*-командой command. Возвращается указатель на структуру типа PGresult, где сохраняется информация, полученная от СУБД в ответ на запрос. При желании можно отсылать сразу несколько *SQL*-команд в одном запросе, разделяя их ; – точкой с запятой. В этом случае информация, сохраненная в структуре PGresult, будет относиться только к последнему запросу.

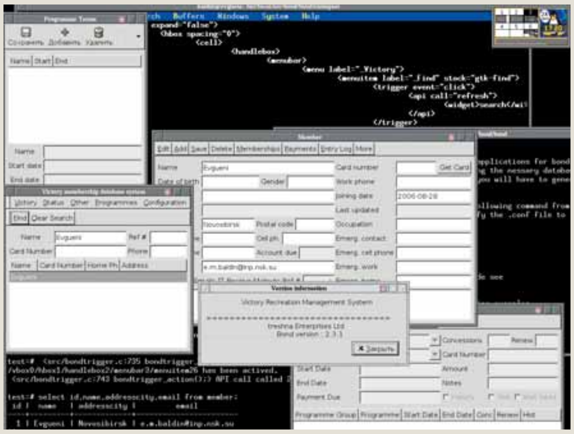
»

[1] Kerberos — промышленный стандарт для аутентификации и взаимодействия в условиях незащищенного окружения. Алгоритмы Kerberos основаны на шифровании с использованием симметричного криптографического ключа и требуют наличия доверенного агента.

bond, но не Джеймс

Лень писать все самому, но не лень изучать XML? Тогда *BOND* – это программа для Вас. Сайт проекта расположен по адресу: http://www.treshna.com/bond/.

Рабочей частью пакета является исполняемый файл **bondfrontend**, который осуществляет связь с базой данных и может «прикинуться» любой формой. Описание формы хранится в обычном XML-файле. Используемый диалект XML подробно описан в документации.



Формочки, XML (правда на заднем фоне и без подсветки) и связь с базой данных — это *bond*.

История пакета насчитывает уже пять лет. Программа доступна по дуальной лицензии: GPL – для использования со свободным ПО и коммерческой – для использования в проприетарном. Существует версия и для Windows.

Перед инсталляцией *bond* следует внимательно прочесть README и установить все, что там перечислено. Сборка осуществляется с помощью scons, который позиционируется как замена *make* с функциональностью *automake/autoconf* и синтаксисом от Python. Установка по умолчанию производится в **/usr/local/**. Для инсталляции (scons install) необходимы привилегии системного администратора. Затем можно приступать к чтению документации и изучению каталога examples.

» По умолчанию, каждый PQexec открывает отдельную транзакцию, если явно не начать ее с помощью команды BEGIN. В последнем случае транзакция будет продолжаться либо до COMMIT, либо до ROLLBACK.

Есть более сложный вызов PQexecParams, который позволяет передавать вызов и параметры к этому вызову раздельно. Таким образом исчезает необходимость самостоятельно формировать строку *SQL*-команды и заботиться об экранировании данных, что важно в случае сохранения бинарных последовательностей. В качестве платы за соображения безопасности PQexecParams способен послать не более одной команды за раз.

В некоторых случаях для увеличения скорости выполнения часто встречающихся запросов полезно обратить внимание на пару функций PQprepare и PQexecPrepared. Эти команды эквивалентны своим *SQL*-аналогам PREPARE и EXECUTE. Идея оптимизации состоит в том, что прежде чем выполнить запрос, *PostgreSQL* сначала анализирует его, затем планирует порядок действий и только потом, собственно, выполняет запрос. Первые два этапа для похожих запросов с разными условиями отбора можно выполнить заранее с помощью команды PREPARE. Затем, с помощью команды EXECUTE, можно выполнять подобные уже подготовленные (prepared) запросы.

Все упомянутые выше команды работают с сервером БД синхронным образом, то есть посылают запрос и ждут ответа. Клиентское приложение на это время «засыпает». В *libpq* предусмотрен целый класс функций, предназначенный для асинхронных операций, не блокирующих клиентское приложение. Их применение усложняет код и логику программы, хотя все в пределах допустимого. С моей точки зрения, лучше организовать все так, чтобы время, использованное на ожидание результатов запроса, не влияло фатально на внешние процессы, и обеспечить бесперебойную работу сети и сервера базы данных.

Информация о состоянии запроса

После выполнения запроса всегда интересно узнать, каково его состояние:

```
ExecStatusType PQresultStatus(const PGresult *res);
```

На вход подается структура PGresult, создаваемая в результате работы PQexec-подобных функций, а на выходе получаем информацию о состоянии в виде числа, значение которого можно сравнить со следующими константами:

- PGRES_COMMAND_OK – все прошло хорошо (для запросов, которые не возвращают данные, например, INSERT),
- PGRES_TUPLES_OK – все прошло хорошо, плюс получены данные в ответ на запрос (для запросов типа SELECT или SHOW),
- PGRES_EMPTY_QUERY – строка запроса почему-то была пустой,
- PGRES_COPY_OUT – идет передача данных от сервера,
- PGRES_COPY_IN – идет передача данных на сервер,
- PGRES_BAD_RESPONSE – ошибка, ответ сервера не разборчив,
- PGRES_NONFATAL_ERROR – нефатальная ошибка: предупреждение (notice) или информация к сведению (warning),
- PGRES_FATAL_ERROR – при выполнении запроса произошла серьезная ошибка.

Для получения более подробной информации об ошибке следует воспользоваться функцией

```
char *PQresultErrorMessage(const PGresult *res);
```

При вызове этой функции в качестве результата будет сформирована строка с информацией об ошибке или пустая строка, если все прошло хорошо.

Получение данных

При получении данных предполагается, что статус запроса соответствует PGRES_TUPLES_OK. Теперь, если примерно известно, что хочется получить в результате запроса, то для получения данных достаточно четырех функций:

```
int PQntuples(const PGresult *res);
int PQnfields(const PGresult *res);
char *PQgetvalue(const PGresult *res,
        int row_number, int column_number);
int PQgetisnull(const PGresult *res,
        int row_number, int column_number);
```

Первые две функции являются информационными и позволяют узнать, сколько строк получено в результате запроса (PQntuples) и сколько в каждой такой строке колонок (PQnfields). Возьмите на заметку, что 0 строк – это тоже хороший результат.

Функция PQgetvalue позволяет получить доступ к данным. В качестве параметров кроме структуры соединения (res) передается номер строки (column_number) и номер колонки (column_number). Все данные возвращаются (как и посылаются) в виде текстовой строки, то есть перед употреблением их необходимо перевести в привычный формат. Например, в случае целых чисел можно воспользоваться функцией atoi.

Следует помнить, что данные *SQL* могут иметь неопределенное значение (**NULL**). Если подобная возможность существует, то перед получением значения следует проверить, определено ли оно. Pqgetisnull позволяет разобраться с этой проблемой. По передаваемым параметрам эта функция эквивалентна PQgetvalue, а в качестве результата возвращает **1**, если значение не определено, и **0**, если определено.

Кроме упомянутых выше, существует целый ряд функций, позволяющих получить информацию о данных, как то: имя колонки (PQfname), размер передаваемых данных в байтах (PQgetlength) и тому подобное. Для экранирования специальных символов при операции с бинарными или текстовыми данными есть набор сервисных функций PQescape*.

COPY

SQL-команда COPY является расширением, специфичным для *PostgreSQL*. Основное преимущество *SQL* – «все есть понятный текст», в некоторых случаях, когда надо передавать большие объемы данных, оборачивается недостатком. Функции PQputCopyData и PQgetCopyData в ряде случаев позволяют значительно ускорить передачу данных между сервером и клиентом.

Асинхронные сигналы

Стандартный *SQL* не предполагает взаимодействия разных пользователей, кроме как через изменение данных в таблицах. *PostgreSQL* позволяет посылать асинхронные сигналы с помощью команд LISTEN и NOTIFY. LISTEN "имя сигнала" передается серверу как обычная *SQL*-команда. Если статус запроса становится равным PGRES_COMMAND_OK, то это означает, что ранее был выполнен запрос NOTIFY "имя сигнала". Если же инициализация сигнала (NOTIFY) ожидается позже регистрации (LISTEN), то функция PQnotifies позволяет вновь проверить наличие сигнала после любого запроса.

Сборка «мусора»

«Мусор» убирать придется руками. Каждая функция типа PQexec создает объект типа PGresult. После того, как вся необходимая информация о результатах запроса получена, следует освободить память, занимаемую этим объектом с помощью команды:

```
void PQclear(PGresult *res);
```

Если утечки памяти вас не волнуют, то можно этого и не делать. В этом случае следует побеспокоиться о том, почему вас не беспокоят утечки памяти?

Большие объекты

Еще один способ сохранять неструктурированные данные в *PostgreSQL* – это сохранять их в больших объектах (Large Objects). *PostgreSQL* предоставляет интерфейс, схожий с файловым интерфейсом Unix: open (lo_open), read (lo_read), write (lo_write), lseek (lo_lseek) и так далее. Все lo_* команды работают со значениями, полученными из колонки с типом oid – это специальный тип данных, который является ссылкой на объект произвольного типа. То есть последовательность работы с большим объектом следующая: создается большой объект (lo_create). Далее возвращаемый lo_create указатель oid используется для записи данных в большой объект (lo_import/lo_write), а затем этот указатель вставляется в таблицу с помощью стандартных SQL-операторов. Чтение происходит в обрат-

ном порядке (lo_export/lo_read). Все операции с большими объектами должны происходить внутри транзакции.

Следует отметить, что необходимость интерфейса больших объектов на текущий момент не так уж и очевидна. Стандартными средствами в *PostgreSQL* можно сохранять бинарные данные размером вплоть до 1 Гб, что вполне может соперничать с максимальным размером для большого объекта (2 Гб).

ECPG

Чтобы не отставать от коммерческих баз данных, *PostgreSQL* имеет свой собственный вариант «встроенного *SQL*». Эта технология позволяет смешивать обычный язык С с SQL-структурами, примерно следующим образом:

```
// файл test.pgc
#include <stdio.h>
#include <stdlib.h>
// структура для обработки ошибок
EXEC SQL include sqlca;
// реакция в случае ошибки/предупреждения
EXEC SQL whenever sqlwarning sqlprint;
EXEC SQL whenever sqlerror do ExitForError();
void ExitForError() {
 fprintf(stderr,"Все, конец - это фатально.\n");
 sqlprint();
 exit(1);
}

int main(int argc, char **argv)
{
// определение переменных, чтобы их можно было использовать
// инструкциях ECPG
 EXEC SQL BEGIN DECLARE SECTION;
  const char *dbname = "test";
  const char *user = "baldin";
  VARCHAR FIO[128];
  VARCHAR NUMBER[128];
 EXEC SQL END DECLARE SECTION;
// соединение с базой данных
// внешние переменные предваряются двоеточием
 EXEC SQL CONNECT TO :dbname USER :user;
// определение курсора через SELECT
 EXEC SQL DECLARE mycursor CURSOR FOR
  SELECT fio, number FROM fiodata,phonedata
          WHERE fiodata.id=phonedata.id;
 EXEC SQL open mycursor;
// чтение данных из курсора
 EXEC SQL FETCH NEXT FROM mycursor INTO :FIO,:NUMBER;
 while (sqlca.sqlcode == 0) { // не 0, если данные больше нет
  printf("ФИО: %s номер: %s\n",FIO.arr,NUMBER.arr);
  EXEC SQL FETCH NEXT FROM mycursor INTO :FIO, :NUMBER;
 }
// разъединение с базой данных
 EXEC SQL DISCONNECT;
}
```

Все SQL-команды начинаются с метки EXEC SQL. Она используется препроцессором ecpg как маркер для конструкций, подлежащих обработке. Внутри SQL-команд можно использовать переменные С. Для этого перед ними необходимо поставить двоеточие «:».

Для компиляции выше процитированного файла (**test1.pgc**) необходимо выполнить следующие действия:

```
> # установить ecpg
> sudo apt-get install libecpg-dev
> # запустить препроцессор
> ecpg test1.pgc
```

»

PostgreSQL в лицах: Сергей Копосов



Визитка LXF:
Дипломированный физик. Профессиональный астроном. Участник программы Google Summer of Code 2006. Домашняя страница: http://lnfm1.sai.msu.ru/~math/.

Евгений М. Балдин (ЕМБ): Что привлекло вас к разработке именно *PostgreSQL*?

Сергей Е. Копосов (СЕК): Вначале я познакомился с *PostgreSQL* как пользователь. Причины были следующие: необходимо было работать с большим астрономическим каталогом (2MASS), который содержал более 200 миллионов звезд. Здесь без базы данных не обойдешься. То, что выбран был именно *PostgreSQL*, связано с тем, что рядом были люди*, которые его уже использовали.

А дальше все пошло по нарастающей. Начав работать с *PostgreSQL*, я вдруг обнаружил, что в нем не существует способов, с помощью которых можно было бы быстро искать объекты на сфере. В итоге это сподвигло меня на создание проекта Q3C (*QuadTreeCube* http://q3c.sf.net). Процесс работы над ним потребовал выйти за рамки стандартного *PostgreSQL*. Например, мне были необходимы bitmap-индексы, а Том Лэйн [Tom Lane] их тогда только реализовывал. Из-за этого мне пришлось перейти на работу с CVS-версией *PostgreSQL*. Во время разработки Q3C мне удалось заметить несколько проблем со скоростью работы bitmap'ов, которые позже были исправлены Томом.

После интенсивной разработки Q3C я сильно «подсел» на списки рассылки *pg-hackers*, *pg-patches,* и я, в частности, написал несколько маленьких патчей для тех проблем в клиенте *pSQL*, которые мне не нравились. Сначала автодополнение DROP~FUNCTION, а потом улучшение работы *pSQL* с многострочными запросами. За последнее меня уже официально включили в список «Official contributors *PostgreSQL*» – чем я несказанно горд [улыбка].

ЕМБ: За что вас выбрали для участия в программе Summer of Code 2006, и чем эта эпопея закончилась?

СЕК: Тут сложно сказать. На самом деле, предложенных на Summer of Code проектов было не так уж и много, а мой проект не был ни чрезвычайно сложным, ни очень простым. Была общая подсказка от Тома, в каком направлении реализовывать проект. Сам проект был вполне естественным улучшением уже существующей функциональности *PostgreSQL*, реализующей к тому же одну из маленьких недостающих частей стандарта SQL:2003. К тому же я уже был хотя бы минимально, но известен в сообществе *pg-hackers*.

А закончилось все тем, что сейчас мой код по агрегатным функциям включен в CVS и будет в версии 8.2 *PostgreSQL*. За что, конечно, спасибо Тому Лэйну, который фиксировал и слегка корректировал код.

ЕМБ: Вы – профессиональный астроном, и, судя по списку проектов, вы активно работаете по основному профилю. *PostgreSQL* не отвлекает от вашей деятельности?

СЕК: Отвлекает. Очень. Поэтому пока я взял паузу в кодировании для *PostgreSQL*. А дальше, поживем – увидим…

* Множество «эти люди» включало Олега Бартунова.

```
> # скомпилировать получившийся файл
> gcc -o test1 test1.c -I`pg_config --includedir` -lecpg
> # проверить работоспособность программы
> ./test1
ФИО: Иванов И.П. номер: 555-32-23
ФИО: Балдин Е.М. номер: 555-41-37
ФИО: Балдин Е.М. номер: (+7)5559323919
```

Удобно это или нет – решайте сами.

Все остальное

Эта статья называется «Интерфейсы», но большая часть ее посвящена только одному из них. Дело в том, что этот один является родным и наиболее полным, а все остальное – лишь подмножество. В простейшем случае все интерфейсы одинаковы: открыл соединение, послал запрос, обработал результаты, закрыл соединение. Также заметна энергосберегающая тенденция везде делать ровно один интерфейс на все типы СУБД.

bash

Да, да к *bash* тоже есть свой интерфейс, правда для этого надо наложить специальный патч. Возни, конечно, немало – зато к базе данных можно будет обращаться прямо из shell-скриптов.

Сайт проекта: http://www.psn.co.jp/postgresql/pgbash/index-e.html.

Java

Совершенно ожидаемо, что Java общается с *PostgreSQL* стандартным образом, а именно через JDBC. Поэтому, если вы знакомы с Java [а если нет – зачем вам использовать ее для доступа к *PostgreSQL*? – прим. ред.], то достаточно добыть драйвер JDBC для *PostgreSQL*, например отсюда: http://jdbc.postgresql.org/. В Debian (Sarge) достаточно набрать

```
> sudo apt-get install libpgjava
```

и, прочитав README к пакету, приступить к работе.

Lisp

Точнее, Common Lisp. Скорее всего, эти драйвера подойдут и для других диалектов:

```
> sudo apt-get install cl-pg
#или
> sudo apt-get install cl-sql-postgresql
```

Второй вариант является драйвером для единого интерфейса доступа к *SQL*-базам данных из *Common Lisp CLSQL* (http://clsql.b9.com/).

Perl

Интерфейс для связи с *PostgreSQL* DBD-Pg используется в Perl через DBI[2]. Все подробности доступны на CPAN: http://search.cpan.org/~dbdpg/dbd-pg/pg.pm.

```
> sudo apt-get install libdbd-pg-perl
```

DBD-Pg охватывает, фактически, все имеющиеся на сегодня возможности *PostgreSQL*: от больших объектов (large objects) до точек сохранения (savepoints).

PHP

О том как использовать *PostgreSQL* в PHP-проектах можно прочитать здесь: http://www.php.net/manual/en/ref.pgsql.php. Драйвер, скорее всего, тоже доступен в репозиториях дистрибутива:

```
> sudo apt-get install php4-pgsql
```

Говорят, почти единственной причиной, по которой PHP-разработчики предпочитают *MySQL*, является то, что раньше не было «родной» версии *PostgreSQL* под Windows. Начиная с *PostgreSQL* 8.0, конкретно этот довод «против» уже не работает.

[2] DBI — унифицированный интерфейс для доступа к данным. Подробнее об этом пакете можно узнать на CPAN: http://search.cpan.org/~timb/DBI-1.52/DBI.pm

Азбука SQL: Г

Управление доступом к данным

Группа операторов *SQL*, ответственных за дифференциацию пользователей путем предоставления и отмены привилегий, представляют из себя специализированный язык управления доступом к данным (Data Control Language).

Привилегии для работу с таблицами базы данных предоставляются пользователям с помощью команды:

```
GRANT «список привилегий»
  ON TABLE  «имя таблицы» [, …]
  TO «пользователь» [, …]
```

Типы привилегий, которые можно предоставить пользователям:

- SELECT – разрешение на выполнение запроса на получение данных,
- INSERT – разрешение на добавление данных в таблицу,
- UPDATE – разрешение на изменение данных,
- DELETE – разрешение на удаление данных,
- REFERENCES – разрешение на создание ограничения по внешнему ключу,
- TRIGGER – разрешение на создание триггера,
- ALL – можно делать абсолютно все.

Эти привилегии можно дать только в случае, если таблица уже существует. На создание таблицы тоже надо иметь право:

```
GRANT CREATE
  ON DATABASE  «имя базы данных» [, …]
  TO «пользователь» [, …]
```

Для отмены уже имеющихся привилегий используется команда:

```
REVOKE «список привилегий»
  ON «имя таблицы» [, …]
  FROM «пользователь» [, …]
```

Python

Модуль для Python существует уже больше десяти лет. Подробности можно узнать здесь: http://www.druid.net/pygresql/. Установка модуля:

```
> sudo apt-get install python-pygresql
```

Ruby

Кое-что можно прочесть здесь: http://ruby.scripting.ca/postgres/. Установка происходит как обычно:

```
> sudo apt-get install libdbd-pg-ruby
```

ODBC

Разработка драйвера идет на pgFoundry. Аскетичная страничка проекта доступна по адресу: http://pgfoundry.org/projects/psqlodbc/. Установка:

```
> sudo apt-get install odbc-postgresql
```

Послесловие

Конечно, наш обзор интерфейсов к *PostgreSQL* нельзя назвать всеобъемлющим, но изложенных сведений вполне хватит для того, чтобы начать работать. Если впоследствии вам придет в голову идея «дописать что-то свое», не спешите хвататься за перо, то есть клавиатуру – советую сначала обратиться по адресу http://techdocs.postgresql.org/oresources.php и посмотреть, что уже сделано. LXF

» Через месяц Настройка PostgreSQL и «1С:Предприятие» на базе открытого ПО

Вёрстка I

ЧАСТЬ 6 Мы учимся использовать культовую систему верстки уже полгода, но до сих пор не сказали об этой самой верстке ни слова. **Евгений Балдин** спешит исправить сложившуюся ситуацию.

Хороший набор – это плотный набор, «дырявый» же набор плохо читается, так как дыры нарушают связанность строки и тем самым затрудняют восприятие мысли.
Ян Чихольд.



Волшебных текстовых процессоров не существует. Телепатией программы пока не обладают. Они, естественно, делают то, что им сказано, но вкус и чувство прекрасного у них полностью отсутствует. В конце занимательного приключения по созданию текстов частенько приходится брать управление в свои руки, дабы навести лоск на почти готовое произведение.

Вёрстка – это процесс составления страниц определённого размера (полос) газеты, журнала, книги из набранных строк, заголовков, иллюстраций и т.п. в соответствии с разметкой или макетом. В этой главе разберёмся с тем как задавать размеры, что такое макет полосы набора и как «удерживать» текст в рамках дозволенного.

Определённые «размеры» и переменные «длины»

Для определения расстояния *LaTeX* поддерживает переменные типа «длина». Например, ранее уже упоминалась команда \TeXtwidth – это переменная, хранящее значение длины, равное ширине текста.

Для создания переменной типа «длина» следует воспользоваться командой \newlength. В качестве обязательного параметра ей передаётся имя переменной. При создании переменной присваивается нулевая длина, так что следующим шагом необходимо приравнять её чему-то:

```
\newlength{\MyLen}
\setlength{\MyLen}{1cm plus 2.5fill minus 5mm}
\addtolength{\MyLen}{5em}
Длина \lstinline!\MyLen! равна \the\MyLen.
```

Длина \MyLen равна 82.89214pt plus 2.5fill minus 14.22636pt.

Длина в *LaTeX* – это не просто какой-то определённый размер. Это более сложная структура с указанием границ возможного сжатия и растяжения. Границы растяжения определяются с помощью инструкции plus, а сжатия – minus. При формировании абзацев *TeX* использует эту информацию для максимально «красивого» заполнения.

Команда \setlength эквивалентна оператору присваивания. В свою очередь команда \addtolength позволяет увеличить переменную на указанную величину, которая может быть отрицательной. Макрос \the позволяет «развернуть» переменную длины для вывода на печать.

LaTeX «говорит» в терминах англо-американской системы мер. Эта система отживает своё, но её наследие будет ещё долго проявляться и портить жизнь современному «метрическому» миру. Для определённости следует знать, что один дюйм (in) равен 2.54 сантиметра, и в нём умещается 72.27 пунктов (1 pt ≈ 0.35 mm). Метрические величины представлены привычными сантиметрами (cm) и миллиметрами (mm). Кроме упомянутых величин, *LaTeX* умеет оперировать размерами в больших пунктах (bp), пунктах Дидо (dd), пиках (pc) и цицеро (cc) – традиционных единицах измерения, используемых в типографиях. Минимальной ненулевой единицей длины в *LaTeX* является приведённый пункт (sp), который составляет 1/65536 от одного пункта.

Кроме определённых единиц измерения, длину можно задавать также и в относительных: 1ex соответствует высоте строчной латинской буквы x, а 1em – ширине прописной латинской буквы M. Эти величины меняются вместе со сменой шрифта, что позволяет задавать автоматически масштабирующиеся горизонтальные промежутки, не привязанные к конкретному размеру и типу шрифта. Например, широкий пробел, задаваемый с помощью команды \quad, определяется как \hspace{1em}.

```
\setlength{\MyLen}{1ex}
Высота x равна \the\MyLen\par
\Large \setlength{\MyLen}{1ex}
Высота x равна \the\MyLen
```

Высота x равна 4.71341pt

Высота x равна 7.43707pt

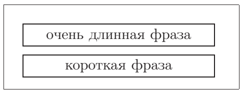
Интересной инструкцией является длина fill – это бесконечность. *TeX* поддерживает операции с бесконечностями, причём оперирует тремя их видами: fil, fill и filll, где fil<<fill<<filll. С помощью этих сущностей производится центрирование боксов и более сложные выравнивания.

Если хочется узнать ширину текста, то можно воспользоваться

Месяц назад Мы использовали LaTeX для печати листингов программ и алгоритмов.

командой \settowidth:

```
\settowidth{\MyLen}{очень длинная фраза}
\addtolength{\MyLen}{1em}
\centering
\framebox[1.2\MyLen]{очень длинная фраза}\par
\framebox[1.2\MyLen]{короткая фраза}
```



Аналогично, команда \settoheight позволяет выяснить высоту текста над базовой линией, а \settodepth – глубину под базовой линией. При использовании длины можно добавить перед ней множитель.

А теперь немного «магии» из английского FAQ по *LaTeX*:

```
\makeatletter
\newcommand{\maxwidth}{%
 \ifdim\Gin@nat@width>\linewidth
   \linewidth
 \else
   \Gin@nat@width
 \fi
}
\makeatother
```

Эта конструкция определяет переменную длины \maxwidth таким образом, что при вставке картинки:

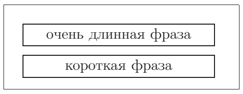
```
\includegraphics[width=\maxwidth]{«картинка»}
```

ширина картинки становится равной минимальному из двух возможных значений: «естественной» ширины картинки (размер в BoundingBox) или ширины строки. Это позволяет вывести картинку в натуральную величину при условии, что она не выходит за рамки дозволенного и загнать её в эти рамки, коли она за них вылезает.

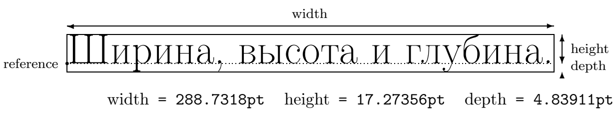
calc

В дополнение к стандартным возможностям, пакет *calc* расширяет базовые операции с длинами. Фактически *calc* вводит арифметические операции в привычной со школы инфиксной записи.

```
\setlength{\MyLen}{
  (1em+\widthof{очень длинная фраза})*\real{1.2}}
\centering
\framebox[\MyLen]{очень длинная фраза}\par
\framebox[\MyLen]{короткая фраза}
```



При загрузке *calc* \setlength и \addtolength переопределяются так, что в качестве аргумента после этого можно передавать арифметические выражения. Кроме арифметики, в *calc* определяются макросы \widthof{текст}, \heightof{текст} и \depthof{текст} – ширина, высота и глубина текста.



› **Определение ширины (width), высоты (height) и глубины (depth).**

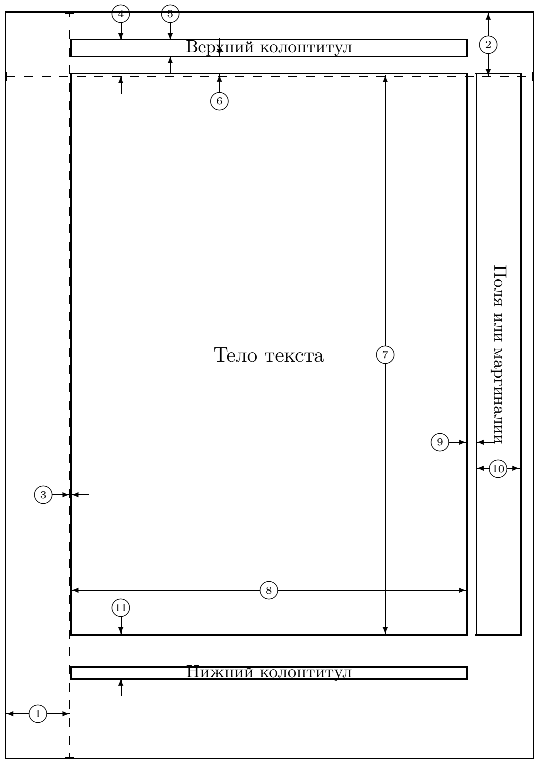
При умножении длины на число длина должна стоять до числа (4mm*2 – верно, а 2*4mm– нет). Делить и умножать можно только на целые числа. Действительные числа вводятся с помощью уже использованного в примере макроса \real и отношения длин, вычисляемого с помощью команды:

```
\ratio{«длина»}{«длина»}
```

Подробное описание пакета можно найти в документации **calc.pdf** из коллекции *tools*.

Скелет страницы

На рисунке ниже приведён результат выполнения команды \layout из одноимённого пакета. Основное место на странице занимает текст – верстальщики зовут его «основной текст» или «тело текста» (bodytext). Справа и слева от текста расположены поля. Поля обычно остаются пустыми, но иногда они используются для заметок («маргиналий» или «фонариков»). В верхней и нижней части страницы расположены, соответственно, верхний и нижний колонтитулы. Колонтитул представляет из себя справочную строку, помогающую ориентироваться в структуре текста.




```
 1  один дюйм + \hoffset         2  один дюйм + \voffset
 3  \oddsidemargin = 2pt         4  \topmargin = -41pt
 5  \headheight = 18pt           6  \headsep = 21pt
 7  \textheight = 635pt          8  \textwidth = 448pt
 9  \marginparsep = 12pt        10  \marginparwidth = 49pt
11  \footskip = 50pt                \marginparpush = 6pt (not shown)
    \hoffset = 0pt                  \voffset = 0pt
    \paperwidth = 597pt             \paperheight = 845pt
```

› **Макет полосы набора класса *scrartcl* (опция a4paper). Результат выполнения команды \layout.**

Совокупность размеров и расположений указанных полей, а также вид и содержание колонтитулов называется макетом полосы набора. На рисунке пунктирной линией изображены поля драйвера (1 и 2) относительно которых выстраиваются все остальные поля. По договорённости отступы до полей драйвера равны одному дюйму. Переопределив \hoffset и \voffset (по умолчанию они равны нулю), можно легко сдвинуть полосу набора целиком по горизонтали и вертикали, соответственно.

Ниже перечислены параметры, которые управляют макетом полосы набора:

» Тело текста характеризуется высотой \TeXthight (7) и шириной \TeXtwidth (8) При многоколоночной вёрстке ширина колонки равна \columnwidth. Переменная \linewidth принимает значение, равное длине строки текущего текста.

» \oddsidemargin (3) добавляется слева в случае односторонней печати. При двусторонней печати полосы набора для чётных и нечётных страниц различаются. В этом случае для нечётных слева опять же добавляется \oddsidemargin, а для чётных \evensidemargin.

» Верхний колонтитул располагается на расстоянии \topmargin (4) от поля драйвера, имеет высоту \headheight (5), а тело текста отступает от колонтитула на расстояние \headsep (6).

» \footskip позиционирует базовую линию нижнего колонтитула относительно последней строки текста.

» Поля для заметок имеют ширину \marginparwidth (10) и отступают от тела текста на расстояние \marginparsep (9). Ещё одна опция управляет минимальным расстоянием между заметками: \marginparpush. »

Выбор размера бумаги

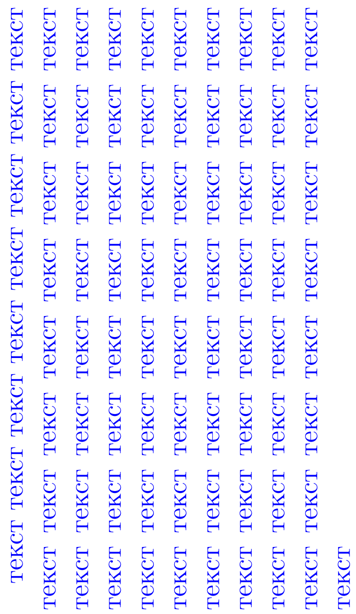
Физический размер бумаги описывается параметрами \paperwidth и \paperheight. Стандартные базовые классы *LaTeX* (article, book, report и letter) по умолчанию предполагают, что для печати используется бумага формата letter. Очевидно, это умолчание не годится для России, где стандартом является формат А4 (210x297 мм). Обычно установить правильный формат можно с помощью передачи параметра a4paper при выборе класса документа:

```
\documentclass[a4paper,12pt,oneside]{scrbook}
```

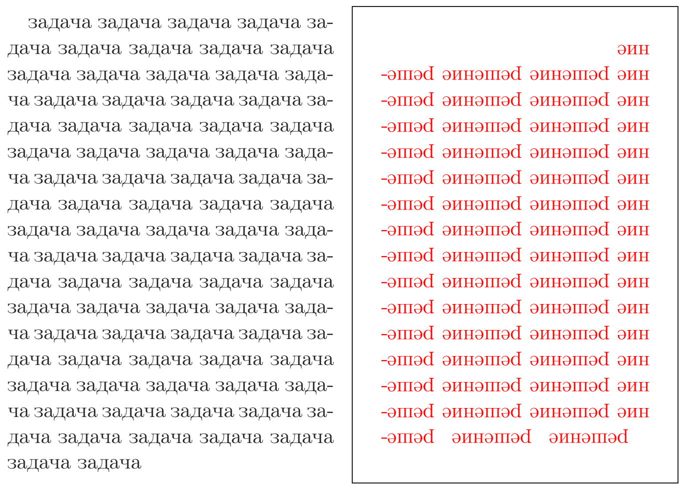
Для создания небольших брошюрок со страницей размера А5 (половина размера А4) используется опция a5paper.

Ориентация

Для портретной и альбомной ориентации вообще-то требуются разные макеты полосы набора. Альбомная ориентация может использоваться по умолчанию в некоторых специализированных классах (например, slides).



› *lscape* в действии.



› *rotpages* в действии.

Если повернуть требуется только текст без изменения полей колонтитулов, то проще всего воспользоваться окружением landscape из пакета *lscape*. Всё, что находится внутри этого окружения поворачивается на 90 градусов против часовой стрелки. При использовании *pdflatex* для изменения ориентации не только текста, но и страницы (дабы не наклонять при чтении текста голову набок) при загрузке пакета следует передать ему опцию pdftex или воспользоваться пакетом-надстройкой *pdflscape*.

Лучше не менять параметры макета во время набора, но если очень хочется это сделать, то можно воспользоваться стандартным пакетом *portland*. Он позволяет менять портретную ориентацию на альбомную и обратно на лету, переопределяя соответствующие управляющие макетом переменные. Команды \portrait и \landscape работают как переключатели между этими режимами. В пакете определены и одноимённые с переключателями окружения. Это довольно низкоуровневый пакет, и пользоваться им надо аккуратно.

Если по какой-то причине требуется развернуть текст на 180 градусов, можно воспользоваться пакетом *rotpages*. В пакете определены два переключателя режима \rotboxpages и \endrotboxpages, которые указывают границы повёрнутого теста. Пакет умеет разворачивать не только страницы, но и колонки. Подробности можно найти в документации: **rotpages-doc.pdf**.

Меняем макет

Причина, по которой неискушённый ТеХник начинает менять макет полосы набора, обычно заключается в том, что он использует стандартные классы *LaTeX* – один из четвёрки article, book, report или letter. Это очень древние классы, и на них много чего «завязано», поэтому они прибывают в абсолютно замороженном состоянии. Поэтому лучше для начала найти себе класс по душе. Например, scrartcl, scrbook, scrreprt и scrlettr – классы из коллекции *KOMA-Script*, которые дублируют функциональность стандартной четвёрки, но ориентируются на европейских пользователей и размер листа А4. А.И. Роженко в рамках коллекции *ncclatex* (NCC) предоставляет класс ncc, ориентированный на русских ТеХников.

Двигаем размеры

Макет лучше не менять!

Выбор, естественно, не ограничивается упомянутыми выше классами – он огромен. Поэтому, прежде чем что-то изменять, следует внимательно изучить уже имеющиеся решения. И даже если вы знаете, что делаете, лучше не переопределять переменные, управляющие размерами макета полосы набора, вручную. Правильным решением будет воспользоваться одним из уже имеющих специализированных пакетов, например: geometry или vmargin.

geometry

Пакет geometry меняет размеры прямо в процессе загрузки стилевого файла, например, так:

```
\usepackage[height=25cm,a4paper,hmargin={3cm,2cm}]{geometry}
```

Разбор параметров выполняется с помощью пакета *keyval*, который уже упоминался в статье «Графика» (см. LXF86). В качестве параметров можно передавать выражения, если загружен пакет *calc*.

Список воспринимаемых пакетом опций очень обширен. В документации (**geometry.pdf**) подробное описание всех имеющихся опций занимает свыше пяти страниц текста. Использование этого пакета для изменения размеров полей является более предпочтительным, чем изменение размеров напрямую. Обилие принимаемых параметров позволит задать любой формат полосы набора в наиболее естественной форме, не ошибившись при расчётах. Подробное описание пакета, к сожалению, выходит за все разумные ограничения на объём статьи. К счастью, документация очень хорошая и чрезвычайно подробная.

vmargin

Пакет *vmargin* управляет размерами макета через выставку полей. Перед этим следует задать формат листа:

```
\setpapersize[«ориентация»]{«формат листа»}
```

В качестве обязательного параметра команды можно указать один из

следующих форматов: A0, A1, …, A9, B0, B1, …, B9, C0,…, C9, USletter, USlegal и USexecutive. Необязательный параметр может принимать значения landscape (альбомная ориентация) или portrait (портретная ориентация, по умолчанию). Для нестандартных форматов листа можно задать размеры с помощью ключевого слова custom команды:

```
\setpapersize{custom}{«ширина»}{«высота»}
```

После того, как вы определитесь с размером страницы, можно задавать размеры для полосы набора с помощью одной из следующих команд:

```
%Полоса набора с колонтитулами
\setmargins{«ширина поля слева»}{«высота поля сверху»}%
        {«ширина текста»}{«высота текста»}%
        {«высота»}{«отступ»}%верхний колонтитул
        {«высота»}{«отступ»}%нижний колонтитул
%Полоса набора без колонтитулов
\setmargnohf{«ширина поля слева»}{«высота поля сверху»}%
        {«ширина текста»}{«высота текста»}%
%Установка рамеров без изменения колонтитулов
\setmarg{«ширина поля слева»}{«высота поля сверху»}%
      {«ширина текста»}{«высота текста»}%
```

Кроме перечисленных команд, определены их аналоги: \setmarginsrb, \setmargnohfrb, \setmargrb. Отличие состоит в том, что вместо ширины и высоты тела текста в качестве параметров им передаются ширина правого поля и высота нижнего поля. Подробно пакет описан в документации **vmargin.pdf**.

Стили страницы

Полоса набора – это не только размеры: это ещё и наполнение колонтитулов. Обычно оно определяется в классе документа. Здесь без необходимости лучше тоже ничего не менять.

Простейший способ изменить стиль страницы – это воспользоваться командой:

```
\pagestyle{«стиль страницы»}
```

Если стиль надо переопределить только для текущей страницы, то следует воспользоваться командой \thispagestyle{стиль страницы}.

Есть три стандартно определённых стиля:

- empty Страница выводится без каких-либо колонтитулов – только текст.
- plain Выводится только номер страницы в нижнем колонтитуле.
- headings В верхнем колонтитуле выводится номер страницы и информация, определяемая классом документа.

Если и это наполнение не устраивает, то можно определить свой собственный стиль. Пакет *fancyhdr* специализируется как раз на этом.

fancyhdr

Чтобы воспользоваться возможностями пакета, необходимо загрузить стилевой файл и выбрать стиль с помощью \pagestyle



Заполняем колонтитулы с помощью *fancyhdr*.

Пакет *fancyhdr* позволяет управлять содержимым колонтитула. Возможности: верхний и нижний колонтитул разбивается на три независимых части, многострочные колонтитулы, колонтитулы, вылезающие по ширине за \textwidth, декоративные линейки, разные колонтитулы для чётных и нечётных страниц, отдельные колонтитулы для специальных полос (начала глав, страницы, отведённые под плавающие объекты). Вот простейший пример для определения колонтитулов с помощью *fancyhdr*:

```
\usepackage{fancyhdr}
  …
\begin{document}
\pagestyle{empty} %очищаем стиль страницы
\pagestyle{fancy} %включаем пользовательский стиль
\lhead{A.B.Top}  %верхний колонтитул слева
\chead{---~\arabic{page}~---} % там же по центру
\rhead{Статья}  %верхний колонтитул справа
% аналогично для оформления нижнего колонтитула
\cfoot{---~\roman{page}~---}
\lfoot{\hspace{0.7cm}Это просто демонстрация}
\rfoot{возможностей \textbf{fancyheadings}}
```

Переменная page содержит номер страницы. Подробности о том что можно сотворить с колонтитулами, следует искать в документации к пакету **fancyhdr.pdf**.

Причёсываем текст

Теперь, когда границы определены, разберемся, что можно сделать для красивого размещения текста на странице. Часть забот по следованию правилам вёрстки *LaTeX* берёт на себя. Вам, например, не требуется следить за единообразием оформления полос. Но есть дефекты, которые компьютеру заметить не под силу, например, так называемые коридоры.

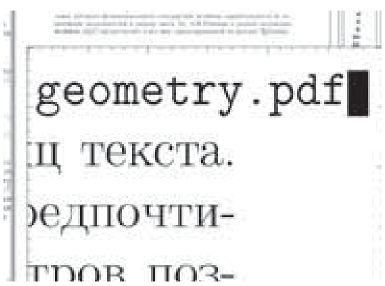
«Коридоры» графически разделяют текст абзаца или полосы на некоторое подобие неожиданных колонок, а они могут осложнять восприятие текста, из-за чего «коридоры» надо устранять как дефект набора.

Дефект набора (коридор).

Строка

Когда строка с точки зрения *TeX* становится слишком разряженной (Underfull) или слишком сжатой (Overfull), в log-файле появляются предупреждения, начинающиеся словами вида:

```
Overfull \hbox (26.1765pt too wide) in paragraph at lines 347--356
[][]\T2A/cmr/m/n/12 Список вос-при-ни-ма-е-мых па-ке-том оп-ций
очень об-ши-рен. В до-ку-мен-та-ции geometry.pdf
Если при выборе класса документа передать ему опцию draft:
\documentclass[draft,a4paper,12pt,oneside]{scrbook}
то такие проблемные места будут отмечаться прямо в тексте.
```



Переполненная строка. Включена опция *draft*.

Обычно подобные переполнения связаны с тем, что *LaTeX* не знает, как перенести какое-либо слово. В этом случае следует сообщить ему, что и где можно переносить, как это было показано во второй части цикла «Базовые элементы» (см. LXF84). В крайнем случае можно насильно разорвать строку с помощью команды \linebreak или \\. В отличии от \linebreak, команда \\ не выравнивает остаток строки по правому полю.

Если можно редактировать текст, то для исправления дефектов набора лучше переделать предложение так, чтобы в новой инкарнации тест не создавал проблем для чтения.

Горизонтальные пробелы

Расстояние между словами можно изменить с помощью горизонтальных промежутков. Они создаются с помощью команды \hspace. В качестве параметра команде передаётся длина. Вариант команды \hspace* отличается от основной тем, что создание пробела не игнорируется даже тогда, когда пробел приходится на начало или конец строки.

Существует несколько определённых по умолчанию горизонтальных пробелов:

- \quad – горизонтальный промежуток шириной 1em. Также есть \qquad – удвоенный \quad, и \endspace – половина от \quad.
- \hfill – бесконечный горизонтальный промежуток. Два \hfill подряд в два раза больше, чем один. Также есть «уменьшенная» бесконечность – \hfil.
- \hrulefill – то же, что и \hfill, но заполненный промежуток подчеркивается. Аналогично, есть команда, заполняющая всё точками – \dotfill.

Страница

Проблемы могут возникнуть и при формировании страниц. В крайнем случае, можно воспользоваться командами принудительного завершения страницы \pagebreak или \newpage. Отличие первой команды от второй в том, что после формирования страницы полоса выравнивается по нижней кромке – это может привести к неоправданному растяжению страницы. Если проблему можно решить путём увеличения/уменьшения страницы на одну-две строки, то лучше воспользоваться следующими макросами:

```
\newcommand{\longpage}{\enlargethispage{\baselineskip}}
\newcommand{\shortpage}{\enlargethispage{-\baselineskip}}
```

Команда \longpage увеличивает тело текста текущей страницы на одну строку, а \shortpage, соответственно, уменьшает. Длина \baselineskip служит для определения интерлиньяжа – междустрочного пробела.

Висячая строка

Одним из самых неприятных дефектов набора является «висячая строка». Висячая строка – это концевая строка абзаца, стоящая первой на странице, или начальная срока абзаца, стоящая на странице последней. Этих артефактов следует всячески избегать. Для подавления этого эффекта в заголовке документа следует определить две переменные:

```
%подавление висячих строк.
\clubpenalty=10000
\widowpenalty=10000
```

Вертикальные просветы

По аналогии с командой \hspace{длина}, вертикальные промежутки организуются с помощью команды \vspace{длина}. Модификация команды \vspace*{длина} создаёт вертикальный просвет, которые не игнорируется, даже если просвет попадает на начало или конец страницы.

Вертикальные просветы также имеют свои умолчания:

- \bigskip – вертикальный промежуток, равный примерно \baselineskip. Также имеются \medskip – половина от \bigskip, и \smallskip – четверть от \bigskip.
- \vfill – бесконечный вертикальный промежуток. Два \vfill подряд в два раза больше, чем один. Также есть «уменьшенная» бесконечность – \vfil.

Печать через две строки

До сих пор временами встречаются требования вида: «Предоставить диплом, набранный в два интервала» – пережиток эпохи печатных машинок [при наборе «в два интервала» интерлиньяж равен удвоенному значению кегля, «в один интервал» – самому значению кегля, – прим. ред.]. Для решение этой проблемы лучше всего воспользоваться пакетом *spacing*. В пакете определена команда \doublespacing, которая выполняет искомое действие. Так же в *spacing* определены макросы \onehalfspacing и \singlespacing – печать в полтора и один интервал, соответственно. Для вертикальной разрядки небольшого фрагмента текста лучше воспользоваться одноимёнными окружениями или окружением spacing:

```
\begin{spacing}{2.5}
  <<Этот текст, напечатан с
  интервалом в две с
  половиной строки>>.
\end{spacing}
```

«Этот текст, напечатан с интервалом в две

с половиной строки».

В качестве основного параметра окружению spacing передаётся интервал, с которым следует печатать текст.

Послесловие

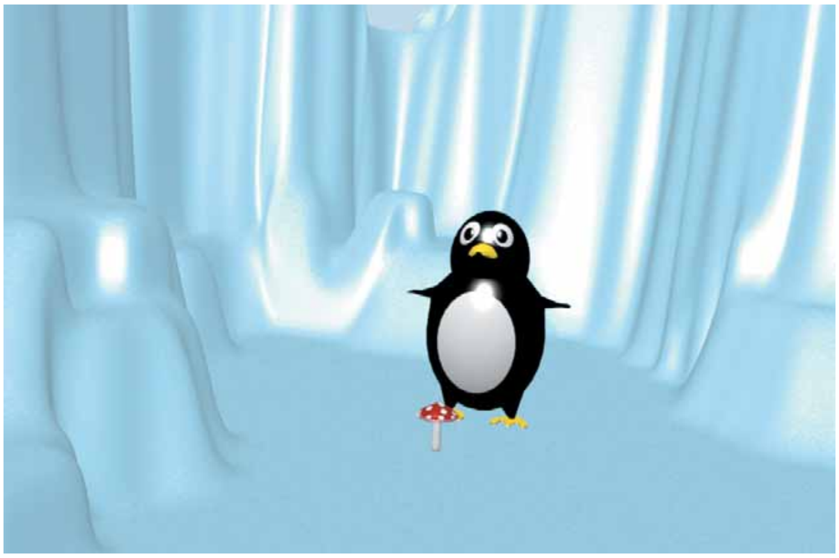
В этом тексте присутствует далеко не вся информация, необходимая для вёрстки текста. Несмотря на то, что *LaTeX* позволяет верстать книги любителям без помощи профессионалов, лучше при любой возможности спрашивать у этих профессионалов совета. Понимание того, что, где и зачем надо исправлять, в случае *LaTeX* чрезвычайно важно, потому что, как правило, и так очевидно. LXF



Через месяц Встречайте путеводитель по классам LaTeX

Моделируем

ЧАСТЬ 2: Интерфейс Blender, по праву считающийся серьезным препятствием на пути к вершинам мастерства, освоен, и сейча самое время создать что-нибудь полезное – например, пингвина. **Андрей Прахов** расскажет, как.



Итак, на предыдущем уроке вы познакомились с интерфейсом *Blender*, попробовали на ощупь модельку пингвина, покрутили, повертели ее. Спасибо за это Александру, а я подхватываю эстафетную палочку и на протяжении всех следующих уроков, мы будем с вами потихонечку осваивать основные инструменты этой мощной программы.

Если вы раньше работали с каким-либо инструментом 3D-моделирования, то многие понятия и возможности будут для вас не в новинку, хотя *Blender* в первую очередь отличается необычностью своего интерфейса. Впрочем, эта часть курса *Blender* уже в прошлом, так что, если вы все еще чувствуете себя неуверенно – советуем держать предыдущий номер *Linux Format* (LXF87/88) открытым. На протяжении оставшихся уроков мы с вами вплотную займемся практикой моделирования и анимирования.

Для начала попробуем создать модель пингвина, наподобие той, что использовалась в первой статье. Место в журнале, к сожалению, ограниченно, и все аспекты и возможности моделинга охватить не удастся. Тем не менее, по окончании этого урока, вы научитесь работать с примитивами, кривыми Безье, материалами. Итак, приступим…

Закройте все ваши прежние наработки и создайте новый, чистый проект (Ctrl+X). По умолчанию, *Blender* сгенерирует для вас куб в Top View. Нам он, естественно, не нужен. Поэтому выделите его правой кнопкой мыши и удалите (X). Коротко напомню, что результаты нажатия на горячие клавиши *Blender* зависят от активного режима. Их несколько, но нас интересуют пока только два: режим редактирования (Edit Mode) и объектный режим (Object Mode), которые переключаются клавишей Tab. Так вот, нужное нам полное удаление объекта происходит только в Object Mode.

Работать над моделью мы начнем с создания головы, которое выполним, используя примитивы. Перейдите в окно Front View (NumPad1). Далее, создайте сферу, нажав клавишу «пробел» и выбрав Add -> Mesh -> UVSphere. Программа запросит количественные характеристики объекта, оставьте все по умолчанию, т.е. (segments=32, rings=32). Давайте сразу договоримся, что в дальнейшем, при создании новых объектов вы всегда будете переходить в Object Mode и снимать все имеющиеся выделения – я не буду специально напоминать вам об этом.

Теперь придадим сфере некоторую яйцевидность. Выполнить это можно, воспользовавшись инструментом масштабирования. Вызывается он клавишей S и воздействует на выделенную область или объект. Напомню, что любые количественные изменения в положении объекта или самого объекта можно выполнять по строго отмеренным порциям. Так, если масштабировать объект при нажатой клавише Ctrl, то изменения будут происходить на 0,1 единицы.

Нажмите S, затем Z (растягиваем по оси Z) и, удерживая Ctrl, дотяните до 1,2000 единиц (**Рис. 1**). Щелчок левой кнопкой мыши зафиксирует результат, правой – отменит.



› Рисунок 1.

Измените название сферы на «Head» (**Рис. 2**).



› Рисунок 2.

Следующий этап – создание глаза. Соберем его из двух сфер: зрачка и белка. Для этого щелкните левой кнопкой в свободном пространстве справа (от себя), тем самым, устанавливая 3D-курсор (именно он служит центром для новых объектов). Создайте сферу: Пробел -> Add -> Mesh -> UVSphere (segments=32, rings=32). Затем сожмите ее (S) по всем осям до 0,3000. Для более комфортной работы в дальнейшем, советую увеличить область экрана поворотом колесика мыши. Сделайте глаз более продолговатым, растянув его по оси Z до 1,2000. Перейдите в «Side View» (NumPad3) и сплющите (S) объект по оси Y («Y») до 0,4000.

Итак, «белок» у нас уже готов, переименуйте его в «EyeAll».

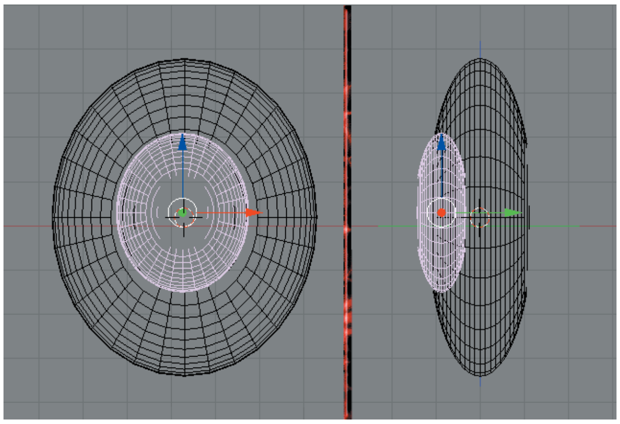
Осталось только приделать глазу зрачок. Особо не мудрствуя, проделаем операцию клонирования «EyeAll». Внимание! Результат клонирования зависит от режима работы программы. Нам нужен независимый, новый объект, только с теми же характеристиками, что и исходный. Для этого переходим во Front View (NumPad1), выбираем режим

ПИНГВИНОВ

LINUX FORMAT на DVD
» Модель Tux'a

Object Mode (Tab), выделяем, если еще не выделен, «EyeAll» и нажимаем Shift+D. Программа генерирует новый объект на том же самом месте, что и оригинал и включает режим перемещения. Переносите объект вправо (от себя) и смените название на EyeSmall». Сожмите (S) по всем осям до 0,5000.

Теперь необходимо совместить две сферы. Это придется сделать вручную, без точных координат. Удобнее всего перейти в режим отображения wireframe (Z) и, переключаясь между видами Front View/ Side View, перемещать объект (G) до получения следующей картинки (**Рис. 3**):



› **Рисунок 3.**

Вот и все, глаз пингвина готов. Для удобства мы не будем сейчас помещать его на место, а займемся созданием клюва. Для этого перейдите в режим Object Mode, снимите все выделения клавишей A и выделите голову. Перейдите в режим Front View и, для удобства работы, отцентрируйте объект клавишей Del на дополнительной клавиатуре.

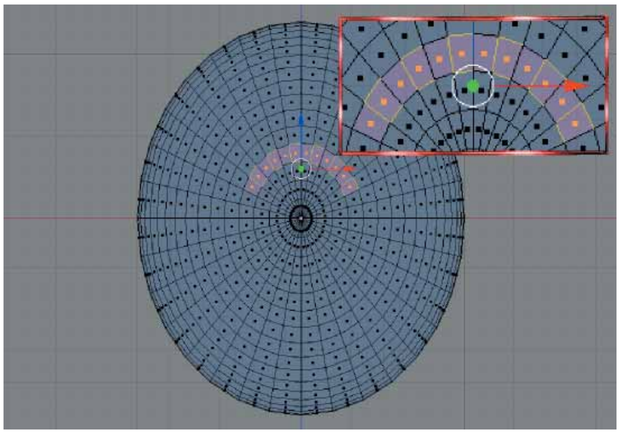
Сейчас нам необходимо, выделив некоторые части объекта, выдавить их в нужном направлении и придать им форму клюва. Для этого перейдите в режим редактирования (Tab) и сбросьте все выделения (A). Так как мы собираемся работать полигонами, активируйте соответствующую кнопку на интерфейсной панели (**Рис. 4**).



› **Рисунок 4.**

А теперь посмотрите на рисунок 5 и постарайтесь воспроизвести его в программе. Для этого щелкните правой кнопкой мыши в нужном месте. Чтобы добавить что-то к уже имеющемуся выделению, удерживайте Shift. Иногда *Blender* ставит отметку не там, где нам бы хотелось, поэтому обязательно прокрутите объект и удостоверьтесь, что не отметили лишнее (**Рис. 5**).

Теперь примените к выделенной части операцию выдавливания. Кстати, extrude – это не масштабирование, при котором происходит изменение свойств объекта, а создание новых подобъектов на основе уже имеющихся. Удобнее всего это сделать в Side View, поэтому перейдите в этот режим (NumPad3). Нажмите клавишу E, выберите

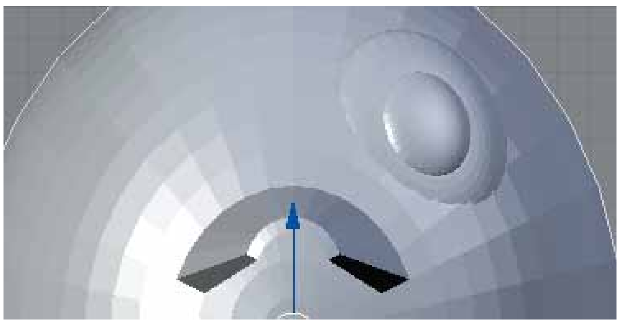


› **Рисунок 5.**

пункт Region из появившегося меню и потяните клюв по оси Y (Y) на -0,5000 единиц.

Получилось что-то, смахивающее на необтесанный нос Буратино, но это поправимо – сделаем его чуток острее и симпатичнее. Выберите просмотр Front View (NumPad1), S для масштабирования и протяните по всем координатам на 0,4000.

Чтобы окончательно разделаться с головой, давайте продублируем глаза и поместим их на законное место. Включите Object Mode и вид окна Front View (NumPad1). Снимите все выделения (A), отметьте обе сферы глаза, нажмите Z для режима отображения wireframe и совместите объекты (**Рис. 6**). Напомню, что для перемещения служит клавиша G, для вращения клавиша R (*см. Урок 1*).



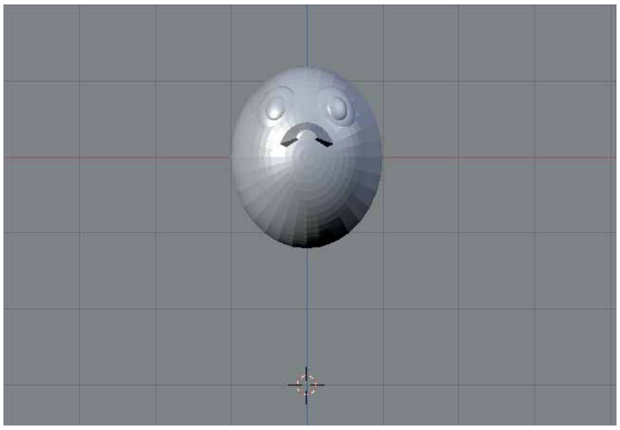
› **Рисунок 6.**

Если операция прошла успешно, то настала пора создать второй глаз. Как вы уже, наверное, догадались, мы опять-таки воспользуемся командой клонирования. Выделите объекты для копирования («EyeAll» и «EyeSmall»), нажмите Shift+D и перетащите их на нужное место. Воспользуйтесь инструментом ротации (R) для точной подгонки глаза. Голова готова!

Туловище

Теперь поработаем над созданием туловища. Для начала подготовим место работы (Object Mode, Front View) и установим 3d-курсор, как показано на рисунке 7. Так как нам необходимо поставить курсор точно по перекрестку линий, воспользуемся меню выравнивания, которое вызывается комбинацией Shift+S и выбе рем из него подменю Cursor-Grid. Если вам не удалось отцентриро- »

» вать курсор, возможно, вначале вы не совсем точно установили его. Дело в том, что программа привязывает курсор к ближайшей сетке. Попробуйте увеличить область, переустановить курсор и повторить привязку.



› **Рисунок 7.**

Туловище будем делать по уже знакомой схеме, из примитива. Создайте сферу с параметрами 32,32 и отмасштабируйте ее по всем осям до 1,5000 единиц. Дополнительно растяните по координате Z на 1,3000 и переименуйте объект в «Body».

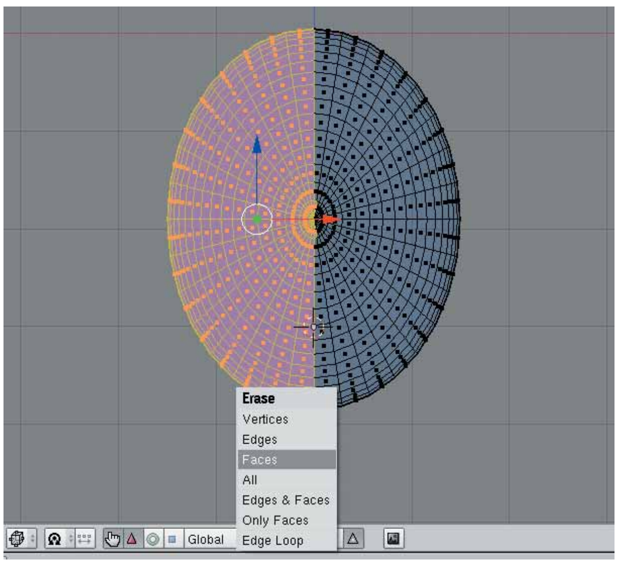
Настало время заняться ластами. У нас (то есть у пингвина) их две, но не будем же мы их делать поодиночке! Разумеется, нет. Давайте создадим одну половину, а потом заставим программу достроить остальное автоматически. Но для начала нам нужно расчистить поле деятельности.

Наравне с известными проприетарными 3D-редакторами, *Blender* также умеет полнофункционально работать со слоями, наподобие *Gimp* или *Photoshop*. Назначение слоев самое различное, но на данном этапе мы просто воспользуемся этой возможностью, чтобы временно убрать некоторые объекты с экрана.

Используйте боксовый курсор (B) для выделения всей головы пингвина и, нажав клавишу M, вызовите панель управления слоями. Выберите левую нижнюю ячейку и нажмите OK.

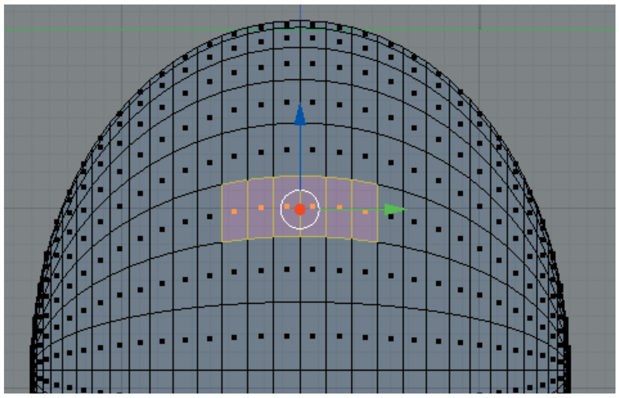
Основная панель управления слоями находится (по умолчанию) внизу экрана. Щелчок мышью по ячейкам вызывает активацию соответствующего слоя, а использование дополнительной клавиши Shift позволяет по-разному комбинировать отражение слоев на экране.

Но вернемся к нашей основной задаче. Сейчас необходимо, воспользовавшись боксовым курсором, аккуратно разделить body на две половинки. Удобнее всего это сделать в режиме wireframe (Z). Не забудьте перейти в режим редактирования! Выделите курсором левую половину строго по центру и удалите ее (клавиша X -> Faces). Внимание: обязательно проверьте, чтобы с правой половины не удалились лишние детали, иначе в дальнейшем возможно появление артефактов (**Рис. 8**).



› **Рисунок 8.**

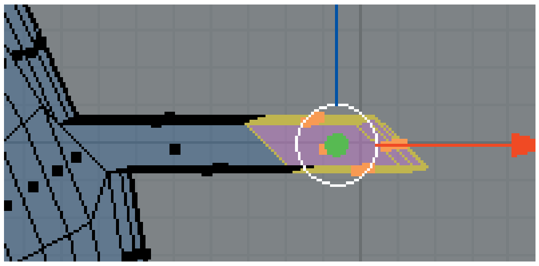
Для выделения полигонов ласты, перейдите в Side View. Отметьте их (**Рис. 9**).



› **Рисунок 9.**

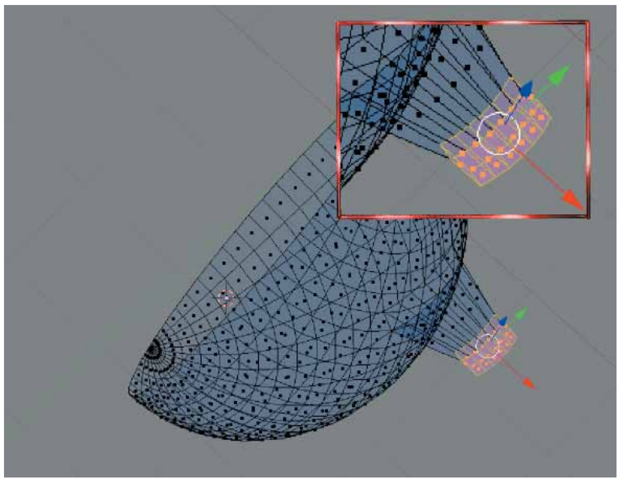
Теперь создайте, с помощью extrude, две ступени ласты. Первая (Front View) по координате X до 0,5000 единиц и вторая, также по X до 0,3000. Выделите всю ласту целиком и отмасштабируйте (S) по Z до 0,4000.

Снимите выделение (A) и отметьте половину ласты, как показано на рисунке 10.



› **Рисунок 10.**

Сожмите (S) по координате Z до 0,4000. Прокрутите объект, как показано на рисунке 11, и немного деформируйте ласту (S) по всем координатам.



› **Рисунок 11.**

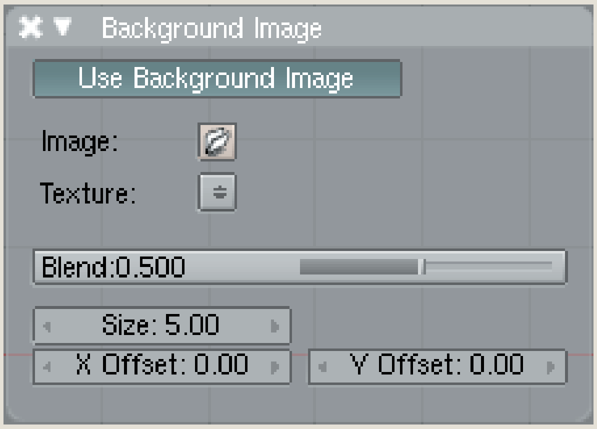
Сейчас нам нужно создать подъем у основания ласты. Для этого нужно выйти из режима wireframe (Z), активировать выделение Vertex Select Mode (*см. начало урока*) и выделить следующую точку (**Рис. 12**). »

Комментарий Сергея Супрунова

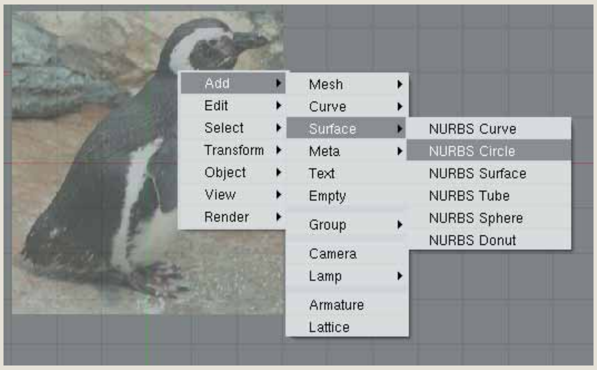


Другие методы создания пингвина: фотореалистичный пингвин

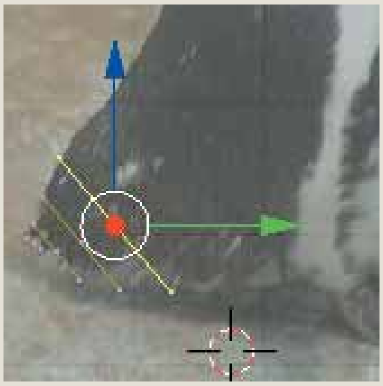
Пингвины бывают разными – это неоспоримый факт. У Андрея получился этакий мультяшный симпатяга; я же выступлю от имени приверженцев реализма. Ну-с, приступим.



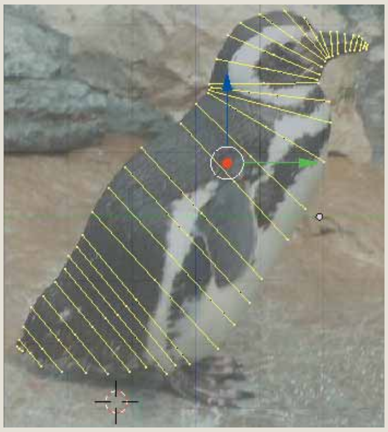
1 Для начала, необходимо найти приличную фотографию пингвина и разместить ее на рабочем столе *Blender*.
Воспользуйтесь меню View > Background Image.
В появившемся окне щелкните на Use Background Image, затем на изображение папки напротив Image. Выберите в диалоговом окне нужную фотографию.
Картинка будет помещена на задний фон, но может оказаться слишком большой или, наоборот, слишком маленькой. Эту несоразмерность можно искоренить, изменяя значение поля Size.



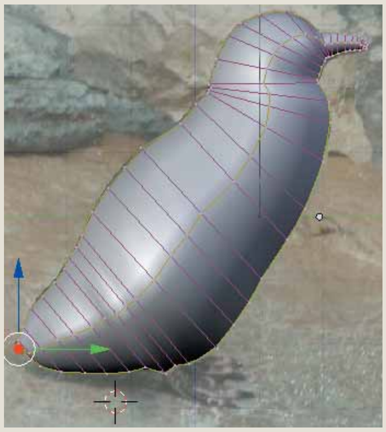
2 Копируя пингвина с натуры, добейтесь желаемой степени реализма.
Переключитесь на вид сверху (NumPad7).
Вызовите меню: клавиша «Пробел» > ADD > Surface > NURBS Circle.
На сцене появится кольцо.
Переключитесь на вид сбоку (NumPad3) и разместите кольцо на самом хвостике пингвина, изменив его размер с помощью Scale (клавиша S) и придав ему соответствующий наклон (клавиша R).
Продублируйте кольцо (SHIFT-D) и разместите его чуть выше (клавиша G).



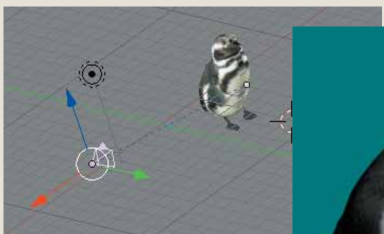
3 Действуя аналогичным образом, «обведите» всего пингвина от хвоста до клюва. Только не трогайте ноги. Ноги и крылья добавим чуть позже.
У вас должно получится нечто подобное тому, что показано на рисунке.



4 Выделите все точки, нажав клавишу «A». А теперь – обычное волшебство: Нажмите клавишу «F», и пингвин обретет плоть!

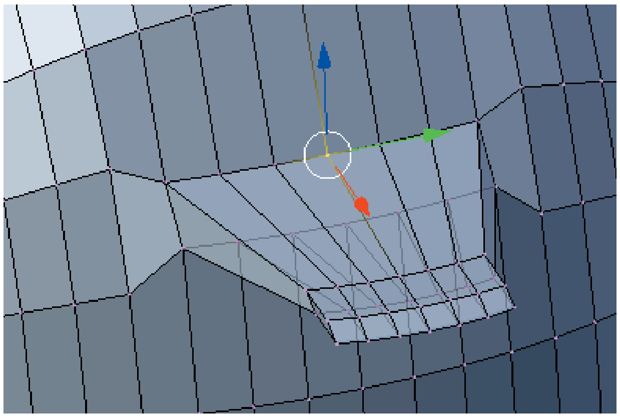


5 Лапы, я думаю, вы уже можете сделать самостоятельно, либо приведенным выше методом, либо экструдированием из сетки (mesh). Для получения более гладких форм рекомендую эффект Catmull-Clark (Subsurf) – его можно найти во вкладке modifier.



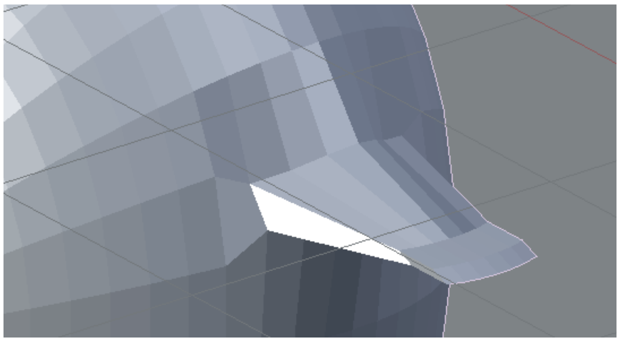


6 Осталось конвертировать пингвина в mesh-объект. Выделите «тело жирное в утесах» в режиме Object Mode и нажмите Alt-C, а чтобы убрать грубые переходы – нажмите кнопку Set Smoot на панели, вызываемой по F9.
Осталось только добавить текстурирование через UV/Image editor, и фотореалистичный пингвин готов. Конечно, каждый готовит его по-своему...



› Рисунок 12.

Теперь, если мы попробуем приподнять эту точку, то у нас сдвинется только она одна и создаст ненужный острый угол. Вот тут-то и пригодится еще один режим редактирования, а именно proportional (пропорциональный). Суть его в том, что вокруг выделенной области создается «мягкая», эластичная зона, которая тянется за выделением. Попробуем создать подъем с использованием этого режима (его включение и отключение производится клавишей O). Итак, включаем режим (O), нажимаем G для движения и пробуем двигать точку. Заметили, что эластичная зона расположена внутри появившегося кольца вокруг выделения? Диаметр кольца можно регулировать колесиком мыши, тем самым отмечая зону эластичности. Используя эту технику, создайте подъем, как показано на рисунке 13.



› Рисунок 13.

Ласта готова. Осталось только приделать вторую половину к туловищу и вернуть голову на место. К счастью, это совсем не сложно. Перейдите в режим Object Mode и во вкладке Modificator интерфейсной панели, найдите кнопку Add Modificator. Выберите Mirror (отражение). А как обратно перенести голову в начальный слой, я думаю, вы и сами догадаетесь, тем более, что подсказка была раньше. Мы же сейчас займемся созданием хвостика.

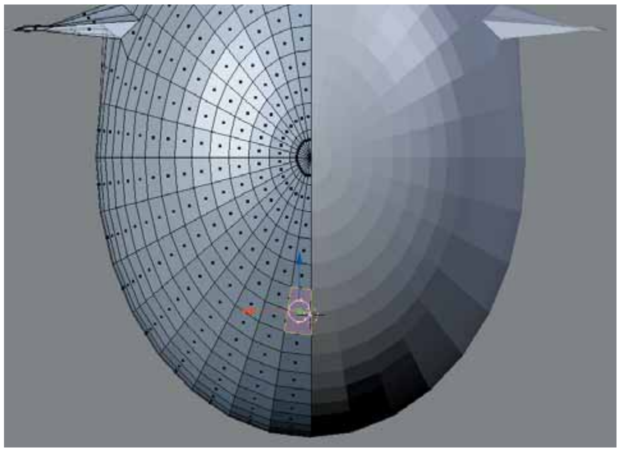
Описание будет очень сжатым, так как все это уже делалось сегодня, и не раз, а впереди у нас еще знакомство с кривыми Безье и работа с материалами. Итак, план следующий:

1. Object Mode, выделить Body, Edit Mode;
2. Прокрутить объект так, чтобы он повернулся к нам задней частью (NumPad4, NumPad6);
3. Отметить два полигона в режиме Face Select Mode (**Рис. 14**);
4. Перейти в Side View и вытянуть E по оси Y до 0,3000;
5. Прокрутить объект обратно, отключить пропорции (O);
6. В закладке Modificator->Mirror активировать кнопку Do Clipping;
7. И, напоследок, сжать хвост (S) по всем осям до 0,3000.

См. рисунок 14.

Кривые Безье

Вот мы и добрались до следующей части нашего урока, а именно, работы с кривыми Безье. Эти объекты используют меньшее количество данных и дают хорошие результаты с меньшим потреблением памяти во

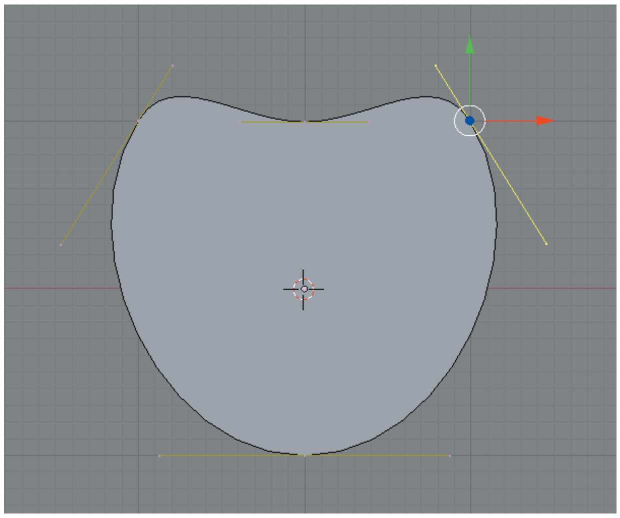


› Рисунок 14.

время моделирования, но забирают больше ресурсов при рендеринге. Кривая состоит из узловых точек, соединенных между собой и управляющих рычагов для каждой вершины. Перемещение рычагов или вершин позволяет нам моделировать кривую, как заблагорассудится. Существует четыре вида рычагов. Наиболее используемые среди них – это Align (по умолчанию, розового цвета) и Free («свободные», черного цвета). Чтобы переключиться между ними, необходимо выделить нужный рычаг или узел и нажать клавишу H.

На данном этапе мы воспользуемся этим гибким инструментом для создания лапки.

Перейдите в Top View (NumPad3) и установите 3D-курсор в пустом, удобном для вас месте. Создайте новый объект – Bezier Circle (Add -> Curve -> Bezier Circle). Не забудьте: если не оговорено иное, подразумевается режим Object Mode. Теперь необходимо, выделяя по очереди верхние узловые точки, перетащить их, как показано на рисунке 15.



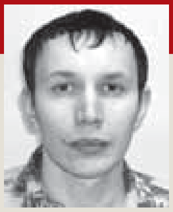
› Рисунок 15.

Добавим между тремя этими вершинами еще две новые. Выделите их, нажмите W для вызова меню редактирования и выберете пункт Subdivide. Перетащите новые узлы так, чтобы они образовали три пальца (**Рис. 16**).

А вот сейчас вам придется, используя приведенную выше информацию, самим доработать внешний вид лапки (используйте рычаги).

То, что у нас получилось, на самом деле, двухмерная фигура. Необходимо придать ей трехмерность. Это можно сделать, конвертировав объект в сетку – mesh. Перейдите в режим Object Mode, выделите лапку и нажмите Alt+C (Mesh). Затем перейдите в режим Edit Mode, выделите полученный объект целиком и выдавите его (E) по оси Z на -0,1000. Сожмите (S) по всем осям до значения 0,4000.

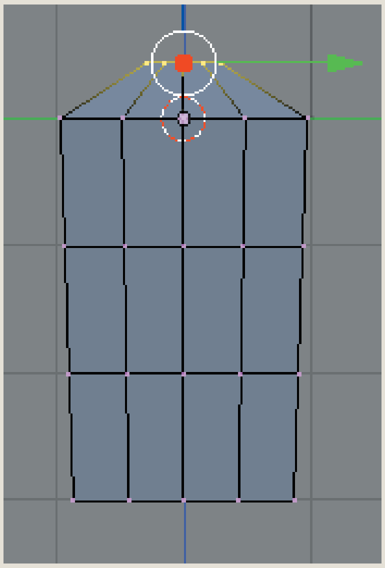
Последнее, что нужно сделать с лапкой – повернуть ее в соответствии с положением туловища. Выполнить это можно вручную или с



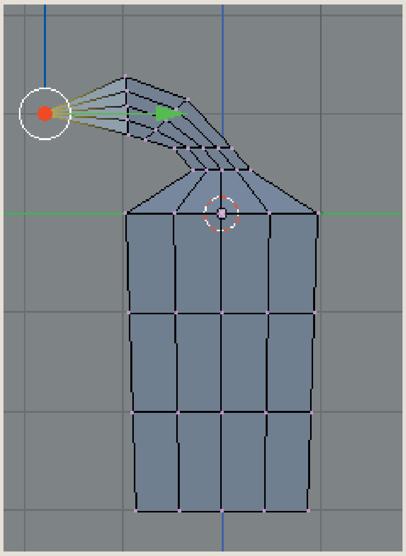
Комментарий Сергея Супрунова

РАЗ ПИНГВИН, ДВА ПИНГВИН

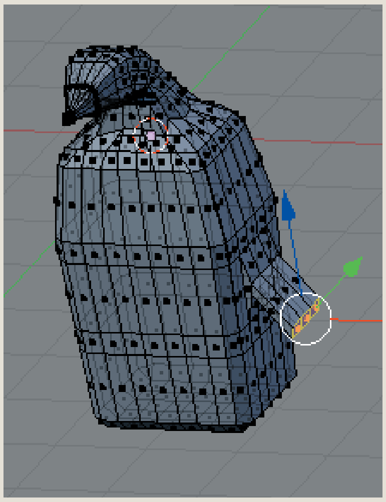
Чтобы создать пингвина – возьмите обычное волшебное полено...
Ладно, я понимаю, что поленья, в отличие от компьютеров, в XXI веке являются дефицитом, поэтому можете просто запустить *Blender*. В центре экрана вы видите куб. Его вполне хватит на то, чтобы смоделировать пингвина.



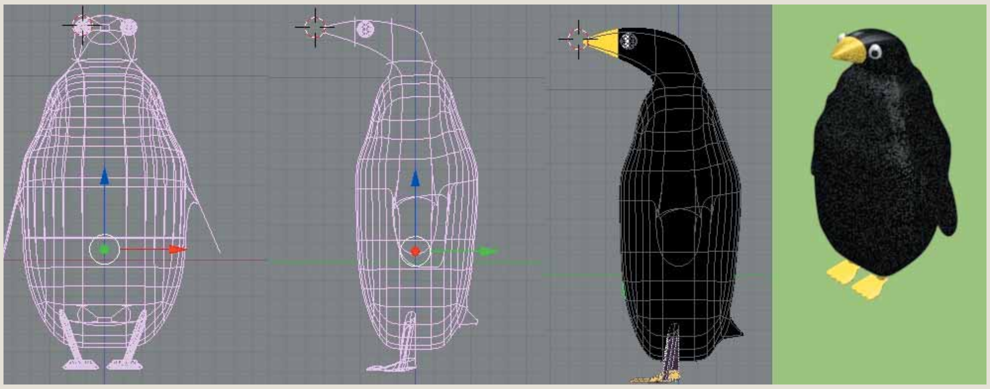
1 Выделите нижнюю часть куба и переместите ее чуть ниже (G).
Сделайте нижнюю часть пингвина уже (S).
Разделите куб на части.
Для этого выделите все вершины, находясь в режиме редактирования (Edit Mode).
Вызовите меню клавишей W и выберите пункт Subdivide.
Дабы почувствовать себя настоящим творцом (или поиздеваться), сделайте это несколько раз.



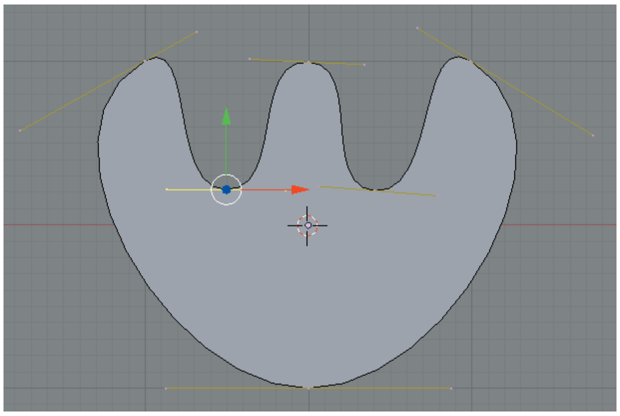
2 Выделите верхнюю часть куба и сделайте ее у́же, используя инструменты Scale (S) и перемещение (G).
Не снимая выделения, переключитесь на вид сбоку (NumPad 3) и экструдируйте вверх, создавая голову. Вам понадобится сделать это несколько раз, немного поворачивая вершины (R), из которых вы вытягиваете голову. В конце заострите клюв, используя инструмент Scale (S).



3 Выделите вершины в области плеч пингвина (мы не будем вдаваясь в дискуссию о том, есть ли у пингвинов плечи – просто выделите вершины, и все) и немного сузьте их (S).
Перейдите в режим Face Select Mode клавишами Ctrl-TAB > Faces и выделите на пингвине плоскости для экструдирования из них крыльев.
Нажмите E и немного вытяните полигон.
Теперь уменьшите его по оси Z (нажмите S, затем Z). Далее экструдируйте крыло и наклоните его вниз (G).
Во вкладке Modifiers > Add Modifier выберите Mirror, и у пингвина вырастет второе крыло.



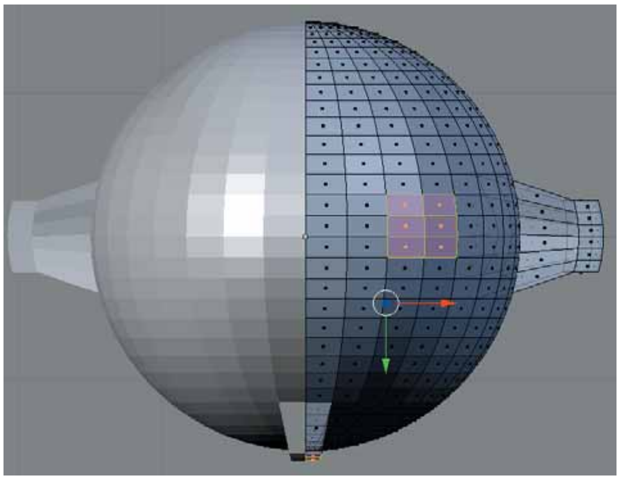
4 Пингвин, как видите, несколько угловат. Скруглим его методом Bevel.
Выделите все полигоны (A).
Нажмите W.
В появившемся меню выберите Bevel. На месте курсора возникнет окно с надписью Recursion. Ничего не меняя, щелкните на OK и тяните курсор к центру.
Пингвин, безусловно округлился. Завершите картину, применив заклятие Armageddon. Простите – Subsurf (Catmull-Clark).
Осталось создать Императору Антарктики ноги и натянуть текстуры.
Что вышло – смотрите на рисунке.



› **Рисунок 16.**

помощью специального инструмента. Выберем последнее. В Object Mode окна Top View (NumPad7) нажмите клавишу N. В появившемся окошке в графе Rot Z укажите значение 180. Переименуйте объект в «Foot».

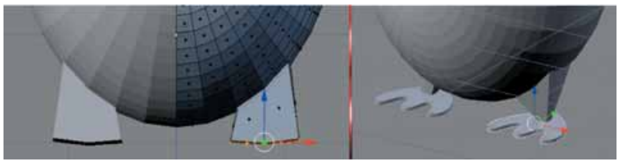
Теперь выдавим ноги из туловища и установим пингвина на лапки. Для этого выделите (в Object Mode) объект «Body» и перейдите в Front View. Прокрутите пингвина до нужного нам места редактирования (NumPad2, NumPad8) и перейдите в Edit Mode. Выделите 6 ячеек, как показано на рисунке 17. Обязательно проверьте, что не выделено ничего лишнего!



› **Рисунок 17.**

Активируйте просмотр Front View (NP1) и вытяните (E) по Z до -0,5000. Нажмите (R) и поверните модель, как показано на рисунке 18. Сожмите (S) по всем осям до 0,2000.

Теперь осталось лишь продублировать лапку и подогнать к полученным ногам. Сделайте это самостоятельно.

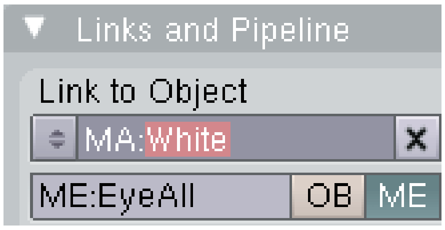


› **Рисунок 18.**

Мы выходим на финишную прямую! Модель почти готова, и теперь можно отдохнуть от бесконечных extrude, scale и т.п. Но это еще не все! Самое интересное впереди. Пора раскрасить пингвина в положенные ему цвета и произвести рендеринг результата.

Раскрасим модель

Переключитесь в режим Object Mode и выделите «Eye big» (белок левого глаза). Нажав F5, вы попадете в панель управления материалами. Найдите в ней вкладку Links and Pipeline и нажмите Add New. Таким образом вы создадите свой первый материал. Назовите его White (**Рис. 19**).



› **Рисунок 19.**

Сейчас нам нужно поменять стандартный цвет материала на белый. Найдите закладку Material и в ней серую полоску образца (напротив col). Щелчком по ней вызывается окно настройки цвета. Естественно, *Blender* позволяет вводить и точные RGB-значения – попозже мы этим воспользуемся.

Заметили, что глаз стал бледнее? Материал применен. Далее выделите правый белок, только вместо создания нового материала, выберите из выпадающего меню уже готовый white (**Рис. 20**).



› **Рисунок 20.**

Как видите, все очень просто. Подобным образом создается черный материал и для зрачков. Давайте назовем его Black и сделаем его более игрушечным, пластмассовым. Самое простое – это поменять характеристики отражения и преломления света. Для этой цели служит закладка Shaders. Введите следующие значения для полей: Ref=0,800; Spec=1,45; Hard=123. Остальное оставьте по умолчанию. Посмотрите, как теперь выглядит материал в окне Preview на интерфейсной панели.

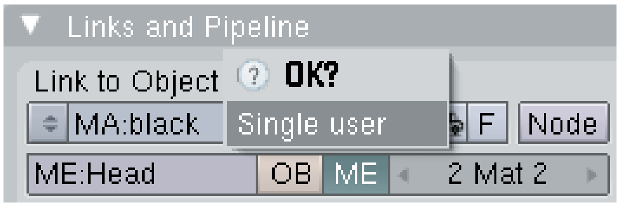
Лапки и клюв у пингвина должны быть желтого цвета. Выделите лапку (Foot) и создайте новый материал (Yellow) со следующими значениями RGB (0,977; 0,911; 0,024), а вот работа с носом – отдельная история.

Создавать и применять материалы к объектам вы уже научились. Но что делать, если нужно применить отдельный материал к выделенной области? К счастью, это не так сложно, как кажется.

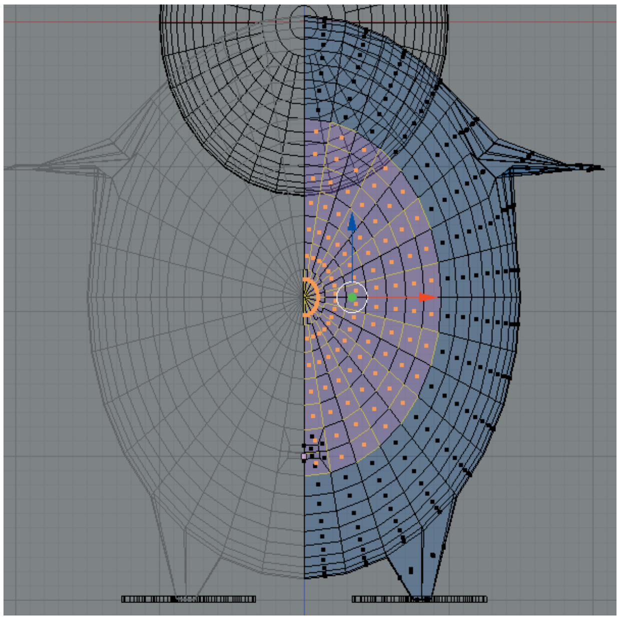
Выделите голову пингвина и примените к ней цвет Black. Перейдите в режим редактирования, боксовым курсором (B) выделите нос и нажмите F9. Таким образом вы активируете режим редактирования объекта. В закладке Link and Material нажмите кнопку New для создания нового материала. Внимание! Основная ошибка начинающих в том, что они пытаются управлять вновь созданным материалом, используя команды группы Vertex Groups (крайние слева). Нам же нужны команды, расположенные рядом и правее.

Верхнее значение 1 Mat 1 поменялось на 2 Mat 2. Чтобы присоединить новый материал к выделению, нажмите кнопку Assign. Выше расположенные кнопки Select и Deselect служат для демонстрации принадлежности выделенной области к активному материалу.

Чтобы поменять Black на Yellow, нужно перейти (F5) в панель материала. В закладке Link and Pipeline обратите внимание на цифру 4, которая находится рядом с названием. Она обозначает количество объектов, которым назначен данный материал. Если вы попытаетесь изменить цвет, то это отразится на всех объектах, что нам, естественно, не нужно. Необходимо разорвать связь с основным материалом. Сделать это можно, щелкнув на цифре 4 и выбрав Single user (**Рис. 21**). Вот и все, теперь вы можете смело выбрать из меню материал Yellow.



› **Рисунок 21.**



› **Рисунок 22.**

Сделайте это.

Осталось сделать брюшко пингвина белым. Как вы уже, наверное, догадались, план работы точно такой же, как и при раскраске клюва. Перейдите в Object Mode, Front View и выделите туловище («Body»). Примените материал Black, нажмите «Tab» для перехода в режим редактирования и в режиме Wireframe («Z») выделите область живота, как показано на рисунке 22.

Ну, а дальше действуйте сами, как при работе с клювом. Мне же пора переходить к заключительной части этого урока.

Осталось сделать пингвина более гладким. Выделите его целиком в Object Mode, нажмите «F9». В закладке Link and Materials нажмите кнопку Set Smooth – модель должна стать сглаженной. Если этого не произошло – проверьте, не выделились ли лишние объекты, типа камеры.

Настало время просмотреть наш результат. Для этого настроим местоположение камеры и источника света. Выделите камеру, нажмите N и в локальных координатах объекта введите: LocX=5,930; LocY=-9,100; LocZ=5,000. Проделайте то же самое и для источника света. Теперь необходимо направить камеру на пингвина. Выделите камеру, затем голову (но не наоборот!) и нажмите Ctrl+T, в появившемся меню выберите Track to Constraint. Вот таким простым образом можно нацелить камеру на нужный объект.

Нажмите F12, и *Blender* покажет вам готовую картинку. Урок закончен! LXF

» **Через месяц** Продолжение работы с материалами и вступление к анимации



Ответы

Есть вопрос по Open Source? Пишите нам по адресу: answers@linuxformat.ru

1 Восстановление разделов

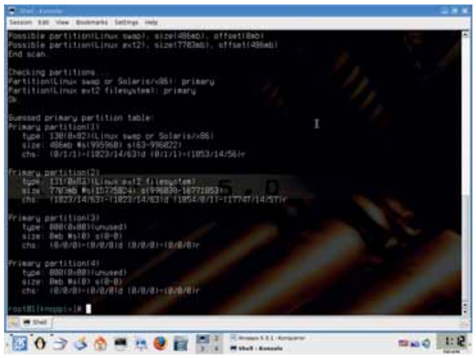
В отчаянных попытках отформатировать новый SATA-диск, я случайно отформатировал другой диск, на котором было три раздела (**/**, **/home** и **swap**). Должен признаться, что пользовался загрузочным диском Windows XP (крайнее средство, честно!). Я нажал кнопку «reset», не сумев остановить процесс по Esc и Ctrl+Alt+Del. Конечно, загрузка с винчестера теперь не идет, но, когда я загружаю *Knoppix* с *QtParted* – мне кажется, что домашний раздел все еще на месте (я вижу значок на рабочем столе), хотя другие разделы накрылись полностью (неформатированное пространство).

Когда я пробую смонтировать раздел (**hda3**) двойным щелчком по его значку на рабочем столе *Knoppix*, возникает ошибка, судя по коду – что-то вроде «файловая система не определена», которая, как я полагаю, связана как-то с первым разделом отформатированного жесткого диска (не там ли хранится «TOC»?). Не поможете?

Mdgreaney, с форума LXF

О Если вы помните размеры разделов, можно создать их заново с помощью *Cfdisk*. До тех пор, пока вы не создадите новую файловую систему на их месте, старая система останется на диске – вероятно, вы удалили лишь таблицу разделов. Может быть, вы не угадаете верный размер разделов с первого раза, но пока будете монтировать их в режиме «только для чтения» (добавляя к команде монтирования **-o ro**), Вы ничего не испортите. Понятно, что с винчестера нельзя загрузиться, ведь корневой раздел удален из таблицы разделов, поэтому *Grub* не может найти свои файлы.

Есть пара программ для автоматизации процесса: *Gpart* (не путайте с *GParted*) и *TestDisk*. Обе присутствуют на *Knoppix 5.0.1* CD и DVD. Учтите, что обе программы будут пытаться угадать расположение разделов по оставшимся данным. man-страничка *Gpart* прекрасно передает суть процесса: «Подчеркиваем, *Gpart* работает эвристически, поэтому никогда не принимайте результаты его работы на веру. Он может быть неопровержимо прав, а может жестоко ошибаться. Вас предупредили».



› В *Knoppix* есть инструменты для восстановления удаленных разделов, например, *Gpart*.

Какую бы программу вы ни выбрали, внимательно и полностью прочтите man-страницу до записи на диск первого байта, и запаситесь терпением. Обе программы работают долго, ведь им нужно просканировать каждый сектор диска. Поэтому пара лишних минут, ушедших на чтение, в общей массе времени будут незаметны, а результат может быть решающим.

Между прочим, TOC (table of contents, содержание) применяется на файловых системах CD и DVD. На жестком диске, в его начале, имеется таблица разделов, информация о директориях содержится в самой файловой системе. **НБ**

2 Да-да, войдите!

В Когда я пробую войти в мой свежеустановленный Ubuntu, выходит сообщение: «Session only lasted 10 seconds» (Продолжительность сеанса 10 сек.).

В файле журнала я вижу следующее:

«Failed to set permission 700 to .gnome2_ private».

Я не могу пройти дальше терминала, как же мне это исправить?

AJB2K3, с форума LXF

О Вы перенесли домашнюю директорию из другого дистрибутива? Описанные симптомы – классические признаки такой ситуации. Даже если вы ввели такое же имя пользователя при инсталляции Ubuntu, имя пользователя и группа могут иметь разные цифровые значения (обычно обозначаются как UID и GID). Файловая система хранит лишь цифровые значения, а значит, эти файлы больше не принадлежат вашему пользователю, отсюда и ошибка при изменении атрибутов.

К счастью, решается все просто и быстро. В терминале наберите следующее:

```
sudo chown -R fred: ~fred
```

Все, что находится в домашней директории *Fred*'а, передастся в собственность *Fred*'у. Команду необходимо давать от имени суперпользователя, оттого она начинается с *sudo*. Замыкающее двоеточие после »

Наши эксперты

» Мы найдем эксперта на любой вопрос! Вы получите ответ на все: от проблем с установкой или модемом до сетевого администрирования; главное – спросить!



Нейл Ботвик

Владелец ISP и экс-редактор дисков для нашего журнала, Нейл считает, что в Linux он от скуки на все руки.



Майк Сондерс

Майк был одним из создателей прототипа LXF – Linux Answers. Его специальности – программирование, оконные менеджеры, скрипты инициализации и SNES.



Стефан Лукас

У Стефана шесть лет администраторского стажа. Теперь он администратор L2 Linux в Rackspace, в его коллекции серверов есть все от Raq3 до Sun Netra X1.



Ник Вейч

В свободное от исчеркивания текстов красными чернилами время Ник возится с Linux-графикой и 3D-приложениями; он у нас отвечает за простые вопросы!

КУДА ПОСЫЛАТЬ ВОПРОСЫ:

Пишите нам по адресу: answers@linuxformat.ru или спрашивайте на форуме: www.linuxforum.ru

» имени пользователя (естественно, вы замените его своим) существенно: оно устанавливает группу владельца равной той группе, к которой принадлежит *fred*. Это не только быстрее, чем запуск *chgrp*, но также устраняет необходимость поиска конкретной группы. **НБ**

3 Двойное видение

В У меня *SimplyMepis 6.0*. Есть звук и Интернет, я доволен *apt-get*, *Beagle* и *SuperKaramba*… все прекрасно. Не хватает лишь одного – 3D-ускорения. Я установил драйверы ATI и активировал их в контрольном центре Mepis, запускал также *aticonfig*, но вот все, что я получил от *glxinfo*:

```
root@1[philippe]# glxinfo | grep direct
_______________________________________________
Xlib: extension "XFree86-DRI" missing on display
":0.0".
direct rendering: No
OpenGL renderer string: Mesa GLX Indirect
```

Прилагаю мой **xorg.conf**

Philippe, с форума LXF

О Крупные конфигурационные файлы вроде Вашего затрудняют поиски проблемы – случай, когда за деревьями леса не видно. Удаление всех строк и секций комментариев упрощает поиск и показывает, что у вас два определения Device для вашей графической карты: одно использует драйверы ATI, а другое – VESA. Это нормально,



› **Programs like *Google Earth* need good 3D acceleration, so check your *X.org* configuration.**

дублирование секции Screen сводит переключение между двумя вариантами к смене строк в секции ServerLayout. Однако эта часть настройки у вас явно нарушена:

```
'Section "ServerLayout"
Screen 0 "ATIScreen" 0 0
Screen 0 "aticonfig-Screen[0]" 0 0
InputDevice "Keyboard0" "CoreKeyboard"
InputDevice "PS/2 Mouse" "CorePointer"
EndSection'.
```

Вы включили два определения для Screen 0 в ServerLayout. Может оказаться, что *X.org* использует первое, так как здесь применяется определение VESA. Но простая проверка файла журнала с

```
grep Screen /var/log/Xorg.0.log
```

покажет, какой экран на самом деле в работе. Для получения 3D-ускорения надо всего лишь закомментировать строку ошибочного определения Screen в ServerLayout и перезапустить X.

Строку можно было и удалить, но закомментировать лучше: Вы в любое время сможете вернуться к драйверу VESA, просто переставив комментарий на другую строку. **НБ**

4 Перенос дистрибутива

В У меня двухдисковый компьютер: на – Windows 98 SE, а на **hdb** – SUSE 10.1 вместе с Mepis 6.0. Я решил сделать основным дистрибутивом Mepis. Все уже инсталлировано на него, и я хочу убрать SUSE, а Mepis переместить на **hda**.

Сейчас загрузка идет через *Grub* от SUSE, установленный в MBR на **hda**, поэтому, удалив SUSE, я потеряю возможность загружаться вообще (исключая Live CD). Нет ли простого способа, вроде полного копирования диска, чтобы перенести Mepis со всеми настройками на **hda**? К сожалению, Win98 вынужден сохранять.

Baron, с форума LXF

О *Grub* установлен в MBR, поэтому он не будет удален вместе с SUSE. Вы потеряете директорию **/boot/Grub**, в которой находятся нужные *Grub*'у файлы, но ее можно заменить такой же от Mepis. Процедуру лучше всего провести с Live CD – Вы ведь не хотите потерять файловые системы, которые ОС может изменить во время копирования.

Загрузитесь с Mepis Live CD и войдите как суперпользователь, с паролем "root". Допустим, SUSE стоит на **/dev/hda2**, а Mepis – на **/dev/hda1**. Сделайте следующее:

```
mount /dev/hda2
mount /dev/hdb1
cp -a /mnt/hda2/boot /mnt/hdb1/boot.suse
```

Этим Вы смонтируете файловые системы и создадите резервную копию загрузочной директории SUSE, она может пригодиться Вам в дальнейшем.

Теперь можно форматировать раздел SUSE. Вам будет предоставлен выбор из нескольких файловых систем, но если Вы не уверены – просто наберите:

```
mke2fs -j /dev/hda2
```

для создания файловой системы ext3. Теперь копируйте все подряд командой:

```
rsync -ax /mnt/hdb1/ /mnt/hda2/
```

Замыкающие слэши обязательны!

Процесс займет некоторое время. Когда он закончится, отредактируйте **/mnt/hda2/etc/fstab** и замените все ссылки на **hdb** новыми, на **hda**. Раздел подкачки SUSE на **/dev/hda** можно оставить без изменений.

Последний шаг – убедитесь, что Ваша система грузится корректно. Если на диске Mepis была директория **/boot/Grub**, нужно отредактировать файл конфигурации **/mnt/hda2/boot/Grub/menu.lst** для смены расположения. *Grub* нумерует диски и разделы от нуля, поэтому **/dev/hda2** «по *Grub*-овски» будет **(hd0,1)**. Может понадобиться добавить пункт меню для загрузки Windows – его можно скопировать из **/mnt/hda2/boot.suse/Grub/menu.lst**. Если же директории *Grub* в Вашей Mepis-инсталляции не было и всем руководил загрузчик SUSE, скопируйте директорию **Grub** из **boot.suse** в **boot** и отредактируйте **menu.lst** с добавкой необходимых пунктов, например: »

Краткая справка о...

Псевдонимы оболочки

Как настроить сокращения для часто используемых команд.

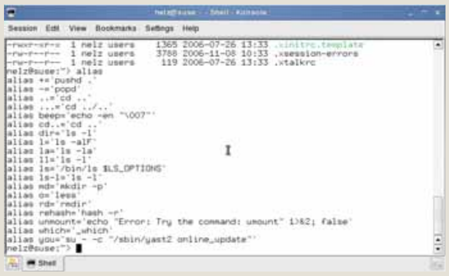
Автозавершение (*см. Ответы*, LXF87/88) сберегает время набора команд и имен файлов, но еще нужно помнить и аргументы, или обращаться за ними в man. Если вы запускаете программу всегда с одними и теми же аргументами – не проще ли установить их «по умолчанию»? А может быть, даже дать одной и той же команде разные названия в зависимости от аргументов? Все это, и еще многое, возможно при использовании псевдонимов оболочки.

Чтобы создать псевдоним (alias), наберите:

```
alias la="ls -lhA --color=auto"
```

Теперь, когда вы наберете la (list all, показать все) в текущей оболочке, вы увидите список с цветным выделением, подробной информацией и отображением скрытых файлов. Введенный вами псевдоним заменяется соответствующей ему строкой перед выполнением его оболочкой.

Но когда вы открываете новую оболочку, вашего псевдонима может там и не быть: в каждой оболочке работает своя команда *alias*. На вид здесь работы еще больше, чем с запоминанием команд, но ее можно автоматизировать. К некоторым файлам оболочка обращается при каждом запуске. Общие для всей системы настройки содержатся в **/etc/profile**, там-то дистрибутив и хранит свои псевдонимы. Затем каждый пользователь может настраивать собственные псевдо-



› **Применение псевдонимов очень облегчает жизнь. Пользуйтесь командой alias.**

нимы в конфигурационных файлах своей оболочки. В случае с *Bash* они хранятся в **~/.bashrc** или **~/.bash_profile**. Первый файл используется во время работы с интерактивной оболочкой, например в окнах *Konsole* или *Xterm*, а второй – во время работы оболочки в качестве «входной» (login shell). Обычно псевдонимы добавляются в **~/.bashrc**.

Псевдоним может выполнять более чем одну команду за раз, например:

```
alias foobar="foo --foo-opts ; bar --bar-opts"
```

Таким способом можно автоматизировать простые последовательности команд без обращения к скриптам.

```
title MEPIS at hda2, kernel 2.6.15-26-386
root (hd0,1)
kernel /boot/vmlinuz-2.6.15-26-386 root=/dev/hda2
nomce quiet vga=791
```

Название файла ядра должно соответствовать находящемуся в Вашей загрузочной директории.

Наконец, запустите *Grub*, чтобы убедиться в верности настройки:

```
grub
root (hd0,1)
setup (hd0)
quit
```

NB

5 Windows убила мой Linux

Во время инсталляции SUSE Linux Enterprise Desktop [SLED] 10 я нарезал свой диск на следующие разделы:

- FAT32, 10GB, /**windows**/C.
- Linux, 10GB, /.
- FAT32, 7GB, /**windows**/E.
- swap, 9GB, swap.

Затем я установил Windows на **C**:. Теперь у меня проблема: Linux не загружается. Я не вижу меню с выбором системы для загрузки, сразу же стартует Windows. Во время установки Windows было такое сообщение: «найден неизвестный раздел, он может быть неактивен, если вы хотите активировать его…», но когда я попытался следовать инструкциям, раздел не был опознан, и теперь у меня Windows лишь на двух разделах, **C**: и **D**: (замечу, что в **D:** только 7 ГБ, а не 10 ГБ).

Что можно сделать?

Penguin, с форума LXF

Это распространенная проблема, вызванная тем, что инсталлятор Windows не проверяет наличие «не-майкрософтовских» операционных систем. Во время установки Windows он переписывает загрузчик по-своему, не считаясь с вашим желанием его сохранить. Хорошая новость – ваша инсталляция Linux осталась нетронутой, включая прежнее меню загрузки и другие настройки. Надо всего лишь переустановить загрузочную запись в Master Boot Record, которую сделал для Вас загрузчик SUSE, *Grub*.

Загрузитесь с SLED CD/DVD и выберите вариант Rescue System из меню; дождитесь приглашения на вход в систему. Наберите **root** (пароль не нужен) – и вот Вы в базовой спасательной оболочке. Первым делом определите, какой из разделов – ваш. Запустите **fdisk –l** для вывода списка разделов. Один из них должен быть обозначен как Linux – скорее всего, **/dev/hda2**, если судить по списку разделов, приведенному выше. Можно смонтировать этот раздел:

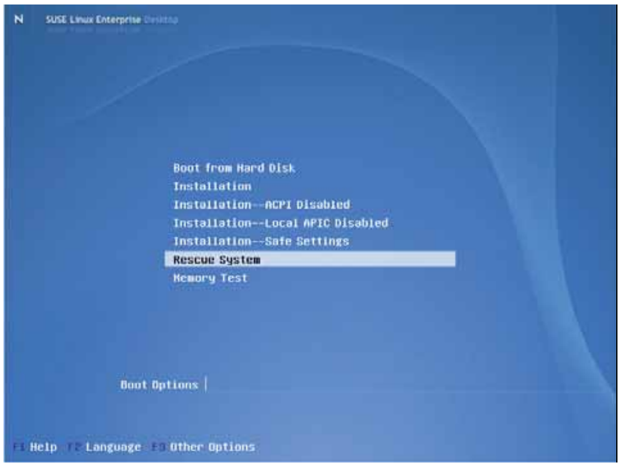
```
mount /dev/hda2 /mnt
```

Затем наберите следующие команды для входа в среду *Grub* и поиска подходящего раздела для загрузчика:

```
grub
find /boot/vmlinuz
```

Будет возвращен загрузочный диск в терминологии *Grub*, скорее всего **(hd0,1)**.

Теперь наберите следующие команды для новой настройки загрузчика:



Режим «спасения» с установочного диска SUSE поможет, если Windows «поправила» ваш винчестер.

```
root (hd0,1) #то, что вернул Вам вызов find
setup (hd0)
quit
```

Вот и все, теперь вы можете перезагрузиться командой *reboot*. Извлеките CD/DVD – и получите свое *Grub*-меню снова, со всеми прежними вариантами. Учтите, что после каждой переустановки Windows будет случаться то же самое, способ лечения – тот же. **НБ**

6 Беспроводной Knoppix

В январе 2006 года вы напечатали интересную статью о создании собственного дистрибутива (Собери свой дистрибутив, LXF74/75). У меня

Часто задаваемые вопросы...

Печать

Знайте, что несложные способы настройки принтеров под Linux существуют!

Что такое CUPS?

Common Unix Printing System (Универсальная система печати для Unix). Это набор драйверов и утилит для полной поддержки, управления и использования принтеров в Linux и других Unix-подобных ОС.

Так это драйвер принтера?

В общем, да, но это намного шире, чем просто драйвер. CUPS помещает «переносимую прослойку» между приложениями и печатающими устройствами. В нее входят и драйверы принтеров, и все необходимое для общения программ с принтерами.

Значит, придется мучиться с командной строкой и файлами конфигурации, чтобы мой принтер заработал?

Вовсе нет. У CUPS есть собственные графические инструменты настройки, работающие через браузер. Зайдите со своим браузером на http://localhost:631 (может понадобиться ваш пользовательский пароль или пароль root) – вы попадете на домашнюю страничку CUPS. Здесь можно просматривать, добавлять и удалять принтеры точно так же, как вы управляете задачами в очереди печати, можно также просмотреть документацию.

А не опасно ли пользоваться браузером вместо более стандартной графической программы?

Сомнения понятны, но стандартная настройка CUPS допускает только подключение к локальной машине. Можно изменить настройку для доступа к локальной сети (Интернет-подключение не рекомендуется), а также для контроля над изменениями, доступными отдельным пользователям.

Что же тогда остается на долю *Gimp-Print*?

Gimp-Print – набор принтерных драйверов, специально созданных для работы с графическим ПО *Gimp*. Хотя *Gimp* неплохо работает и с CUPS, некоторые принтеры дают лучший результат на драйверах *Gimp-Print*. Эти драйверы теперь тоже работают с CUPS, поэтому *Gimp-Print* можно рассматривать как расширенный набор драйверов, работающих через CUPS со всеми программами, не только с *Gimp*.

А что такое *Gutenprint*?

А, это новое название *Gimp-Print*. В зависимости от крутизны вашего дистрибутива вы можете получить как *Gimp-Print* 4x, так и *Gutenprint 5.0*. *Gimp-Print* предназначена теперь не только для Gimp, отсюда и новое название, оно ближе к назначению набора. Но смена названия всегда вызывает – да уже и вызвало – некоторую путаницу.

Как узнать, поддерживается ли мой принтер?

Обратитесь к www.linuxprinting.org. Там содержится внушительная база данных о том, насколько хорошо (или не очень) поддерживается каждый принтер, и приведены советы по использованию драйверов.

Лучше посетить этот сайт до покупки принтера, тогда новая вещь уж точно не окажется бесполезной.

» проблема с беспроводной сетевой картой ноутбука на основе Intel ipw2200 (версия PCMCIA). Ну, я и решил (у меня *Knoppix* 5.01): загружу firmware для карт Intel, помещу его в **/lib/firmware** и подгоню *Knoppix* под это firmware, тогда карту можно будет определить и настроить, или по крайней мере сделать ее настраиваемой во время загрузки Live-дистрибутива.

Я думал, что драйверы для этой карты встроены в ядро, но они не работают без firmware. Могу ли я пользоваться *Knoppix* с Live-дистрибутивом, и настроить при этом беспроводную сеть?

Grumpy@home, с форума LXF

Конечно, можете: когда Вы находитесь в среде *chroot*, описанной в Шаге 3 статьи номера LXF74, можно удалять и инсталлировать любые файлы, точно так же, как в обычном *Knoppix*. Изменения будут внесены в Ваш новый Live CD на Шаге 6.

В ядро *Knoppix* уже встроен драйвер для этой карты, но если Вы решите перекомпилировать ядро, то позаботьтесь о включении параметра CONFIG_IPW2200. Сканер оборудования *Knoppix* найдет карту, загрузит модуль, все настроит и попытается установить беспроводной интерфейс с использованием DHCP. Если Вы хотите, чтобы при загрузке исполнялись определенные команды, поместите их в **/etc/init.d/rc.local** (это имя общеупотребительное, но можно назначить любое другое), затем активируйте вот так:

```
chmod 774 /etc/init.d/rc.local
ln -s ../init.d/rc.local /etc/rc5.d/S99local
```

Команды в директориях **rc** исполняются по порядку, поэтому 99 в имени вынудит команду стартовать последней. Обычно это и требуется, но если Вы хотите перед этим запустить что-нибудь еще – уменьшите номер (или увеличьте номер другой службы). Не вводите слишком маленькие номера. И помните, что Вы работаете с базовой системы, находящейся на CD в режиме «только для чтения», и испортить что-либо почти невозможно.

Если Ваш *Knoppix* не грузится – сделайте шаг назад и попробуйте снова: Вы ничего не теряете.

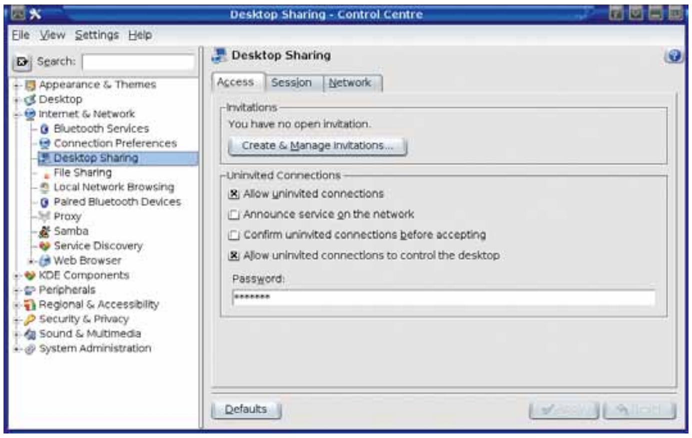
7 Только не на телевизор

У меня дома локальная сеть из пяти компьютеров, на одном из них ни мыши, ни клавиатуры, ни монитора, но есть порт TV-Out, соединенный кабелем S-Video с аналоговым телевизором. В этом компьютере есть карта HDTV, а **xorg.conf** я настроил таким образом, чтобы выход видеокарты клонировался на телевизор. Все работает прекрасно, если я подключаю клавиатуру и мышь и использую телевизор в качестве монитора.

Я могу удаленно подключаться к компьютеру через NX (предпочтительно) или *TightVNC*, но не знаю, как получить доступ к Desktop:0, который выводится на TV. Поискал в Google, но не нашел ничего, что помогло бы решить проблему. По-моему, альтернативное решение – **xorg.conf** для вывода Desktop:1 на телевизор. У меня сейчас Kubuntu 6.10.

Билл [Bill]

VNC предназначен для работы в режиме отдельной X-сессии, но это можно изменить. Запустив *X11vnc*, взятую на www.karlrunge.com/x11vnc, Вы можете сделать существующий X-дисплей доступным через VNC.



› **Клиент и сервер *Rdesktop* в KDE сводят проблему организации удаленного доступа к простому щелчку мышью.**

Но раз уж у Вас Kubuntu, а следовательно, рабочий стол KDE, ответ упрощается: воспользуйтесь сервером и клиентом *Rdesktop*, содержащимися в KDE. На «немом» компьютере запустите Центр управления KDE и войдите в секцию Internet & Network > Desktop Sharing (Разделение рабочих столов).

Теперь установите флажок «Allow Uninvited Connections» (Разрешать подключения без приглашения) и сбросьте «Confirm Uninvited Connections» (Подтверждать подключения без приглашения). Нужно установить пароль для предотвращения неавторизованных подключений. Неплохая идея – блокировать порт 5900 на Вашем маршрутизаторе, если только Вы не планируете подключаться через Интернет. Теперь можете входить через K-Menu > Internet > Krdc Remote Desktop Connection на другом компьютере.

Альтернатива – подключиться к ПК через SSH и запускать отдельные X-программы на Вашем локальном рабочем столе, вот так:

```
ssh -X hostname someprogram
```

При таком решении вывод компьютерной программы не появляется на экране телевизора. Это может быть полезно, если Вы выполняете некоторые административные задачи во время просмотра телепрограмм. Хотя некоторые фильмы определенно украсит окно *Konsole* вверху экрана, не все члены семьи будут согласны с этим.

Настройка *X.org* на первый дисплей не поможет, так как стандартный сервер VNC использует нулевой. **НБ**

8 Переустановка BIOS

У меня четыре совершенно разных компьютера. Шесть месяцев назад я ввел в них одинаковые пароли для защиты *CMOS*. В пароле было шесть букв, знаки * и &. Потом мне понадобилось инсталлировать или изменить часть оборудования, но ни один компьютер не пускает меня в CMOS с этим паролем. Пароль стопроцентно верный. Как же переустановить CMOS?

Питер [Peter]

Скорее всего, пароль содержал недопустимые символы. Способы переустановки CMOS зависят от изготовителя, но чаще всего на материнской плате есть переключатель, помеченный CLEAR CMOS или RESET RTC. Нужно выключить напряжение, перевести этот переключатель в положение переустановки – то есть противоположное текущему положению – подождать несколько секунд, а затем вернуть его обратно.

Ни в коем случае не делайте при включенном питании. Фактически, следует даже вынуть вилку из розетки, чтобы блок питания не подавал на материнскую плату даже малейшего напряжения. Включив компьютер, Вы найдете BIOS полностью освеженным, с первоначальными настройками.

Процедура может различаться в зависимости от модели, поэтому все вышесказанное следует воспринимать как руководство к действию, а не как строгие указания. Одно правило непреложно – питание должно быть отключено, иначе можно повредить материнскую плату. Прочтите руководство по Вашей плате, посетите сайт изготовителя в поисках PDF-варианта, если печатная версия недоступна.

Важнее всего перед началом работы прочесть документацию. Процедура примерно одинакова для всех материнских плат, которые я видел, но детали различаются (одна из виденных мною плат требовала еще и снятия батарейки). **НБ**

9 Последовательный захват

У нас есть небольшой коммутатор 5ESS, который посылает файлы журналов на ROP (read-only printer) на 1200-O-7-1. Кабель – обычный трехпроводной последовательный Unix-кабель. Компьютер соединяется со всем этим через мост Black Box. Система – Windows 2000 с эмулятором терминала *Procomm*.

Я могу непосредственно подключиться к последовательному порту, и файлы журналов выводятся на экран. Я еще перехватываю их и пишу в файл (экономится куча бумаги).

Хотел бы делать то же самое в Linux (чтоб избавиться от Windows), но не могу сделать этого при помощи *tee* или перенаправления. В идеале, мне нужно не только постоянное отображение файлов на экране, но и продолжение их записи в файл даже в случае выхода пользователя из системы. Кроме того, я хотел бы, чтобы каждую полночь создавался новый файл, а старый закрывался. Можете помочь с синтаксисом или подсказать действенный метод сделать это?

Майкл Ивз [Michael Ives]

Я бы первым делом попробовал программу *Minicom*. Это эмулятор терминала, по типу Procomm, он должен быть в репозитории Вашего дистрибутива. Если его там нет, можно загрузить с http://alioth.debian.org/projects/minicom. *Minicom* имеет опцию вывода данных как в файл, так и на экран – обеспечьте ему связь с Вашим коммутатором, и он сделает все, что Вам нужно.

Вопрос-победитель (английская редакция)

Скользкий DMZ

Я установил web-сервер (SUSE 10.0) в виртуальной машине на моем ПК с SUSE 10.0. Настроил его так, что он находится в DMZ [демилитаризованная зона, открытая внешним сетям и почти отрезанная от внутренних, – прим. пер.] моего роутера (тоже SUSE 10.0). Web-трафик направляется в ПК корректно; но я не могу подключиться ни к одному порту из внутренней сети. Хотелось бы непосредственно связываться с ПК по *ssh* из внутренней сети.

Брандмауэр на роутере (192.168.0.9) был настроен с помощью *Yast*. Я организовал проброс порта 80 на адрес web-сервера (192.168.1.2). Я пробовал переназначить серверу внутренний (192.168.0) порт 80, но безрезультатно.

Можно ли решить задачу с помощью *Yast*? Если нет, посоветуйте несложный способ конвертации существующей настройки *Yast* в сценарий *iptables*, где это сделать проще. Надеюсь на вашу помощь…

Ли [Lee]

Настоятельно рекомендую установить IPCop, специализированный Linux-брандмауэр, а не использовать SUSE 10.0 в качестве роутера. Я долго пользовался IPCop, пока не перешел на брандмауэры Cisco PIX: установка займет всего несколько минут. IPCop использует концепцию «зеленой сети» (Green network) для защищенных внутренних сетей, вроде сети Вашего сервера, что упрощает решение поставленной задачи.

Превосходные HOWTO о IPCop можно найти на http://howtoforge.net/perfect_linux_firewall_ipcop_p2, а домашняя страница проекта находится по адресу: www.ipcop.org. **СЛ**

Среди альтернатив – более низкоуровневые соединения с последовательным портом. Например, Logserial с www.gtlib.cc.gatech.edu/pub/Linux/system/serial: он сбрасывает данные с указанного последовательного порта на *stdout* или в файл; в обоих случаях пригодится *tee*. Есть Perl-скрипт на http://aplawrence.com/Unix/logger.html, хотя тут может понадобиться некоторая правка для соответствия вашим требованиям.

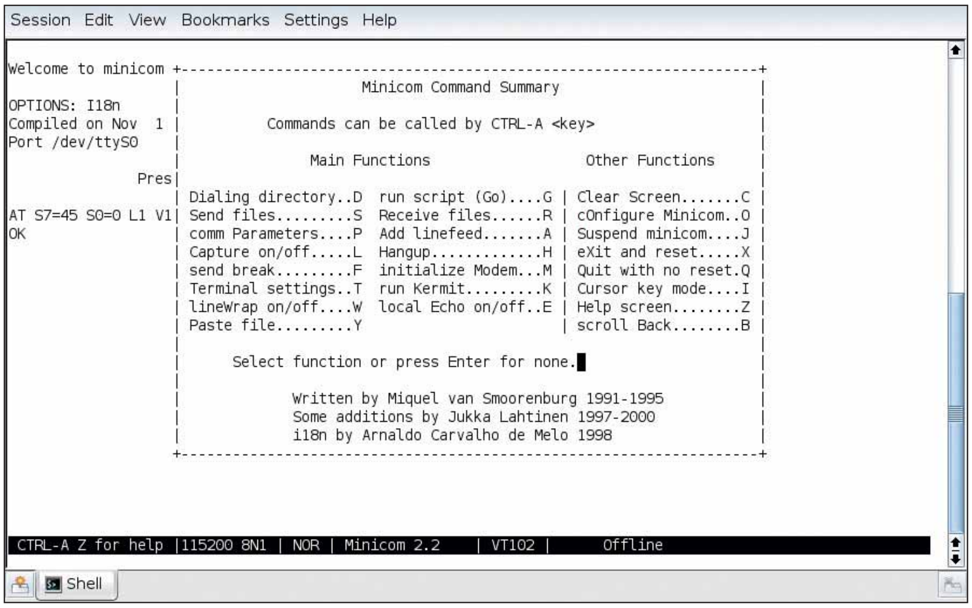
Чтобы программа работала непрерывно, запускайте ее в *screen*. Это сохранит программу рабочей при выходе пользователя, плюс вы сможете подключаться к ней в любое время, даже удаленно через SSH.

Настроив запись журналов и другие параметры в конфигурации *Minicom*, запустите его командой:

```
screen minicom
```

Нажмите Ctrl+A D для выхода из *screen*, оставив *Minicom* работать, затем наберите **screen --r** для вторичного соединения. Необходимо помнить одну вещь: обе программы используют сочетание клавиш Ctrl+A как команду. Чтобы *screen* передавал команду программе, работающей в нем, нажмите Ctrl+A A, и по нажатию Ctrl+A A Z Вы получите экран помощи *Minicom*.

Для разделения файлов журнала я бы воспользовался *Logrotate*. Она устанавливается и работает по умолчанию на большинстве дистрибутивов в качестве ротатора системных журналов.



› **Эмулятор терминала *Minicom* показывает и записывает в файл трафик последовательного порта.**

10 Дайте инструмент…

Мне нужна помощь с послеустановочной настройкой почтового клиента *Evolution* на моей системе Ubuntu. На моем домашнем компьютере двойная загрузка – на одном разделе Windows XP, на другом – Ubuntu Linux. У меня есть Ethernet-подключение к сетевому концентратору. *Mozilla* под Ubuntu работает замечательно – я успешно использую ее для обновлений, поисков в Google и т.п. Поэтому Ubuntu имеет превосходную связь с Интернетом.

А вот почта *Evolution* нуждается в правильной настройке. В одном из старых номеров LXF упоминалось о мастере настройки, но я не смог его вызвать. Поиск в Google выдал документ о настройке *Evolution*. Сперва он казался весьма полезным, но требует меню Tools (Инструменты), которого в моем *Evolution* нет. Как мне настроить *Evolution* без меню Tools?

Джон Клив [John Cleave]

Вы не уточнили, какие именно версии *Evolution* и Ubuntu установлены у вас, но рискну предположить, что новейшие.

Так вот у *Evolution 2.0* было меню Tools, а у *Evolution 2.4* и *2.6* его нет. При первом запуске *Evolution* вы должны были заметить появление First-Run Assistant (Помощника). У вас, похоже, он не сработал, и *Evolution* скрыл свои настройки. Чтобы вызвать Помощника еще раз, нужно удалить директорию **.Evolution** (не пропустите точку!) в домашнем каталоге, затем набрать **Evolution** в командной строке. Тогда *Evolution* решит, что запускается впервые, и вызовет Помощника, с его помощью вы и настроите программу.

Если Вы не хотите связываться с командной строкой, просто создайте новую учетную запись почты щелчками на Edit (Редактировать) > Preferences (Параметры), затем Mail Accounts (Учетные записи) > Add (Добавить). **СЛ**

11 Загадка Kino

Пару дней назад с помощью *Synaptic* я инсталлировал *Kino* на свой Ubuntu 6.06. Кажется, версия *Kino* была 0.8, работала она прекрасно во всех отношениях, включая захват видео с моей камеры.

Сегодня я согласился на предложение автоматического обновления Ubuntu обновить *Kino* до версии 0.9.2, но теперь, запуская *Kino*, я не могу сохранить видео. Вместо этого, выходит сообщение:

'Warning: dv1394 kernel module not loaded or failure to read/write /dev/ieee1394/dv/host0/PAL/in'.

Моя камера отображается как устройство захвата, а *Dvgrab* из *Kino* захватывает все без проблем (только пользоваться им премерзко), поэтому 1394 должно работать.

Есть ли способ как-нибудь это исправить?

bof, с форума LXF

Хм… А Вы не обновляли еще что-нибудь в то же время? Для управления устройствами Ubuntu использует *udev*, и когда *raw1394* перестает штатно работать, оно не создается (эти же проблемы встречаются на Fedora и других дистрибутивах, постепенно переходящих на *udev*).

Это к вопросу «почему». А вот ответ, «что делать»: грубый хак, просто создайте устройство сами:

```
mknod /dev/raw1394 c 171 0
```

Должно сработать. Если нет, проверьте, все ли нужные модули были загружены. Запуск *lsmod* пока- »

» жет Вам, были ли загружены *raw1394* и *video1394*. Может понадобиться смена прав доступа к устройству, чтобы *Kino* могла их прочесть.

Кстати, составители пакетов очень сердиты на упорство, с которым *Kino* цепляется за устройство *raw1394*. Кроме всего прочего, ослабление политики прав доступа вызывает вопросы по безопасности. В будущем *Kino* скорее всего перейдет на более стандартную схему доступа к видеоустройствам. **НВ**

12 Права и… не права

ВЕсли я использую изображения из Сети в своем слайд-шоу, какими правилами регулируются подобные заимствования? Я не нашел никаких ассоциированных сведений относительно авторских прав для тех изображений, которые хочу использовать, но все-таки хотелось бы знать, какие правила существуют для изображений из Интернета.

Аноним, с форума LXF

ОВ Великобритании [и в России, – прим.ред.] авторское право действует с момента создания, и его не обязательно утверждать специально. Поэтому для изображений неизвестных авторов из Интернет лучше считать, что они защищены авторским правом, если явно не указано обратное.

Однако есть и исключения из британских правил, главным образом в случаях исследования или уместного использования (например, если Вы описываете чей-то сайт, правильно будет привести его изображение). Полезную информацию по этому вопросу может предоставить сайт Британского патентного бюро: www.patent.gov.uk/copy/c-manage/c-useenforce/c-useenforce-use/c-useenforce-use-exception.htm.

Таким образом, в некоторых обстоятельствах Вы можете использовать картинки вполне законно, но всегда лучше спросить на это разрешение. Если Вам нужны какие-то изображения для Вашей презентации – могу я предложить Вам работы под лицензией Creative Commons? Зайдите на www.creativecommons.org. **НВ**

Нужна помощь!

› Для наилучшего ответа на ваш вопрос нам нужно знать как можно больше подробностей. Детально опишите конфигурацию системы. Если вы получили сообщение об ошибке, приведите текст сообщения и точно опишите вызвавшие его действия. Если у вас проблемы с оборудованием, то опишите его. Если Linux уже запущен, то выполните в root-терминале следующие команды и прикрепите к письму файл **system.txt**:

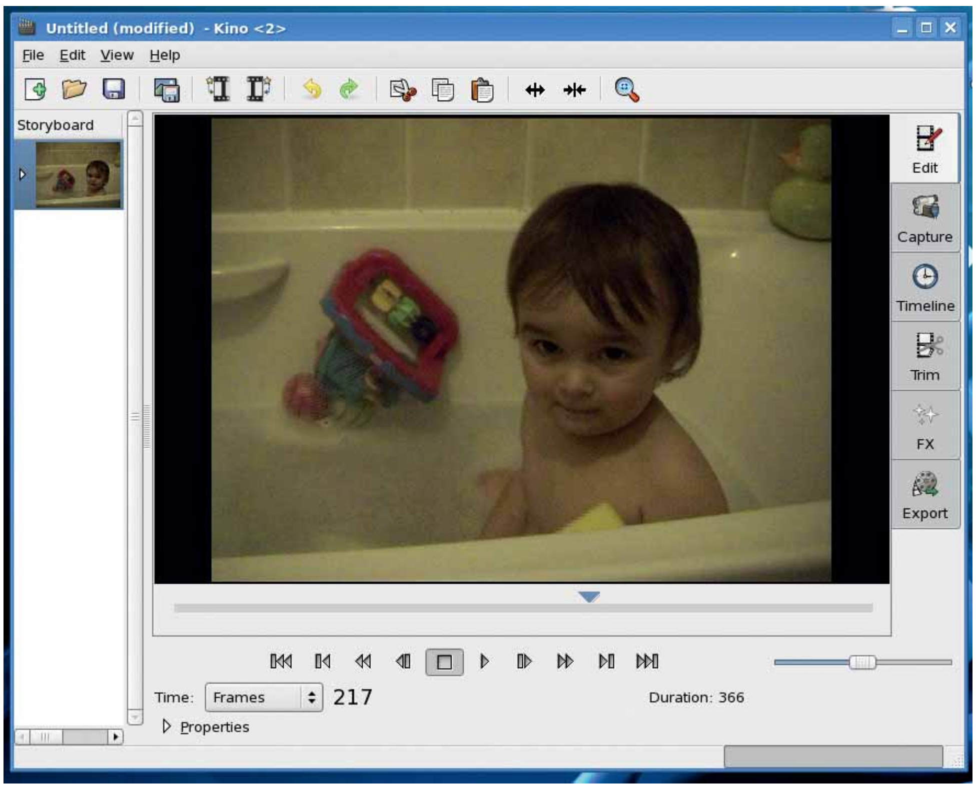
```
uname -a >system.txt
lspci >>system.txt
lspci -vv >>system.txt
```

› Пожалуйста, помните, что сотрудники журнала НЕ являются авторами или разработчиками Linux, любых пакетов или дистрибутивов. Зачастую люди, отвечающие за приложения, выкладывают большую часть информации на web-сайты. Попробуйте почитать документацию!

Мы стараемся ответить на все вопросы. Если вы не нашли ответ на свой, это, возможно, потому, что мы уже ответили на похожий вопрос.



› *Kino* захватывает цифровое видео с ленты или в реальном времени, но только если «видит» устройство.



Лучшие новинки открытого ПО на планете

LXFHotPicks



Ричард Драммонд
Ричард – свободный разработчик, писатель и отец двух детей. Он живет в Индиане, США, где отчаянно скучает по британскому ТВ, подогретому пиву и сосискам.

В ЭТОТ РАЗ ТОЛЬКО ДЛЯ ВАС: NSPluginWrapper » Goggles » Hardware Lister » FLPhoto » Frozen Bubble » Enigma » JMemorize » Atop » CuteCom » Dissy

Расширение браузера

NSPluginWrapper

Версия 0.9.90.4 **Сайт** http://gwenole.beauchesne.info/projects/nspluginwrapper

Представьте: вы только что сменили компьютер x86 на сияющий новый AMD64. Счастливые дни… и вдруг наступает облом. Издатели проприетарных расширений для браузеров не поддерживают вашу платформу. А значит, нет больше Flash, *RealPlayer* или Java-апплетов. Без Radio 4 и YouTube жить явно не стоит!

AMD64 может запускать двоичные файлы x86, но 64-битные программы не умеют взаимодействовать с 32-битными библиотеками. Поэтому ваш родной браузер не может использовать подключаемые модули x86. Один из выходов – использовать *NSPluginWrapper*, хитроумное решение от сотрудника Mandriva и маэстро эмуляторов Гвеноля Бошеня [Gwenole Beauchesne]. Для каждого x86-расширения браузера устанавливается 64-битная обертка, которая позволяет вашему родному браузеру взаимодействовать с x86-расширением через RPC (удаленный вызов процедур) посредством сокетов Unix. *NSPluginWrapper* не завершен, но показал хорошую совместимость с расширениями *Adobe Reader*, *Flash Player* 7 и *RealPlayer* 10, и вы можете использовать их в браузерах на базе *Mozilla* и *Konqueror*'*е*.

Для Mandriva есть RPM *NSPluginWrapper*, но если вам необходимо собрать его из исходных текстов, то это может оказаться сложновато, поскольку пакет содержит 64- и 32-битные компоненты. Если ваш дистрибутив не является биархитектурным (как Mandriva), то вам следует компилировать часть x86 отдельно; попробуйте для этого создать специальный x86 chroot. 32-битный модуль, конечно же, требует своей собственной среды для запуска, так что на вашей машине необходимо наличие набора 32-битных библиотек.

NSPluginWrapper включает командную оболочку для создания и манипулирования 64-битными обертками. Следует отметить, что обертки хранятся в **/usr/lib64/mozilla/plug-ins**. Если у вас установлен Firefox, он может использовать другой путь, и вам придется вручную скопировать обертки и в этот каталог.



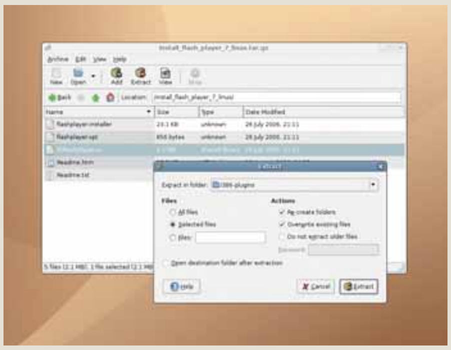
» **Лучший довод за использование расширения Flash – возможность играть в web-игры!**

NSPluginWrapper можно также собрать с использованием эмулятора Qemu. Пока фирма Adobe не убедится, что есть на свете процессоры и кроме x86, *NSPluginWrapper* может значительно улучшить ваши возможности web-серфинга.

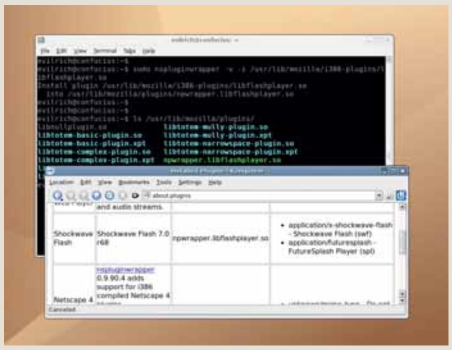
Шаг за шагом: Установка расширения



» **Получите Flash!**
Загрузите версию 7.0 (не 9.0) Linux *Flash Player* с **www.adobe.com/products/flashplayer**.



» **Распакуйте**
Извлеките файл **libflashplayer.so** из загруженного архива и установите от имени суперпользователя в **/usr/lib/mozilla/i386-plugins**.



» **Заверните**
От имени суперпользователя, выполните *NSPluginWrapper* **-i /usr/lib/mozilla/i386-plugins/libflashplayer.so**. Перезапустите ваш браузер и проверьте **about:plugins**.

DVD-проигрыватель

Goggles

Версия 0.91 **Сайт** www.fifthplanet.net/goggles.html

В Linux есть множество приложений, позволяющих проигрывать DVD, но такие проекты часто теряют фокус: они гонятся за функциональностью, а не за удобством просмотра. *Goggles* не таков. Он делает только одно дело, зато делает его хорошо: и это просмотр DVD.

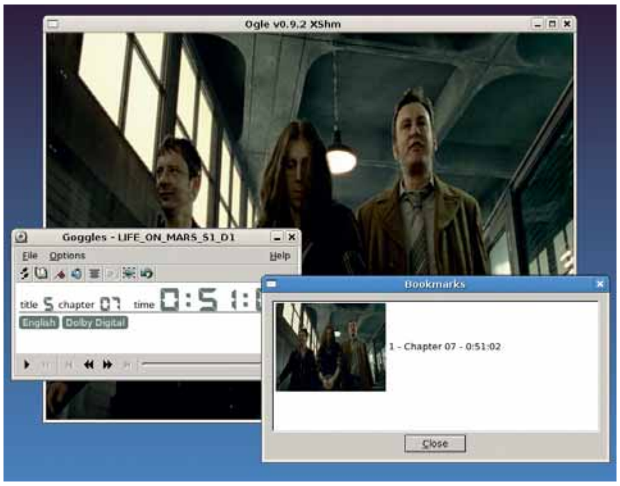
На самом деле *Goggles* – графический интерфейс для DVD-проигрывателя *Ogle*. Он был написан автором инструментария Fox и представляет приятный интерфейс, имитирующий LCD-дисплей реального DVD-плейера. Он ярок, имеет минимум визуальных элементов для выполнения задач вручную и интуитивно понятен в использовании.

Ogle, рабочая часть *Goggles*, был первым DVD-проигрывателем для Linux, поддержившим DVD-меню, и хотя новых релизов не было уже три года, он все еще держится бодрячком в сравнении с более активно развивающимися проигрывателями. *Ogle* работает с меню прозрачнее, чем конкуренты, и единственная функция, которой ему не хватает, это поддержка чересстрочной развертки. Он использует расширение *XVideo* для ускорения отображения видео на вашем экране, но может работать и без него – учтите, что если вы решите обойтись без *XVideo*, то масштабирование и коррекция соотношения сторон будут недоступны.

Комбинация *Goggles*/*Ogle* имеет другие приятные функции. Можно автоматически записать вашу сессию при выходе, и в следующий раз продолжить просмотр DVD с места остановки. Имеется также превосходная система закладок, позволяющая вам помечать позиции, к которым вы хотите вернуться позднее. Закладки сохраняются вместе со снимком экрана в данной точке, наглядно показывая, в каком месте заложена закладка. Закладки сохраняются для каждого диска.

Автор, Сандер Янсен [Sander Jansen], предоставляет двоичные пакеты только для Arch Linux, но сборка *Goggles* проста. Вам потребуется установить инструментарий Fox версии 1.6 (см. www.fox-toolkit.org). *Goggles* избегает вездесущей системы Autotools, используя вместо нее собственные скрипты сборки. Просто выполните **./gb** в корне древа исходных текстов *Goggles* для конфигурирования и сборки приложения, и **./gb install** для установки. Поддерживаются обычные опции настройки, например, **--prefix** для указания пути установки. Выполните **./gb** help для получения списка всех опций. Как мы и говорили, запуск *Goggles* не столь трудное дело, а потрудиться стоит. Следите за проектом.

> «Goggles не такой, как все: он делает только одно дело, зато делает хорошо.»



В окне плейера – *Жизнь на Марсе: Часть 1* с канала BBC

› Благодаря поддержке восстановления сессии и закладок в *Goggles* ваше любимое место в фильме больше никогда не потеряется.

Утилита для оборудования

Hardware Lister

Версия B.02.09 **Сайт** www.ezix.org/software/lshw.html

При отсылке отчета об ошибке, особенно если она связана с драйвером устройства, часто бывает полезно приложить детальные сведения о вашей системе. Вручную вводить такую информацию неудобно, поскольку это требует времени и грозит ошибками. И хотя Linux включает различные стандартные инструменты для извлечения информации об устройствах – например, вывод **/proc/cpuinfo** дает информацию о процессоре, *lspci* выведет список имеющихся в системе устройств, *lsusb* – USB-устройств, и так далее – неплохо иметь один инструмент, выполняющий всю их работу и создающий отчет в удобной форме. Угадали, о чем мы? Да, это *Hardware Lister*, или, кратко, *Lshw*.

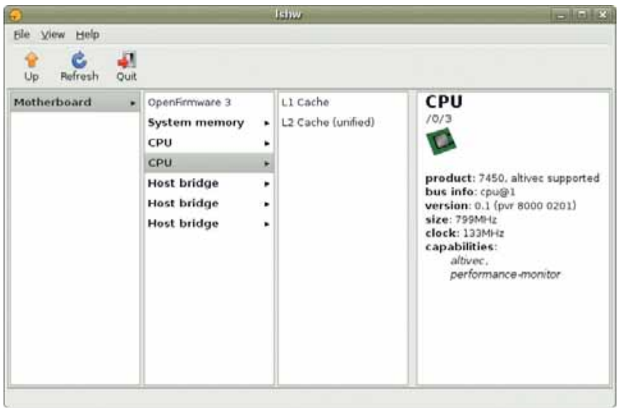
Lshw может вывести отчет обо всех аспектах вашей конфигурации, включая материнскую плату, процессор, память, PCI-устройства, диски и прочее. Он работает на множестве платформ, включая x86, AMD64, PowerPC, Alpha, PA-Risc и Itanuim с ядрами серий 2.4 и 2.6. На системах x86 он поддерживает Desktop Management Interface-enabled BIOS (управление BIOS с рабочего стола, материнские платы моложе шести лет должны поддерживать эту технологию) и EFI. Также поддерживается Open Firmware (через стандартную иерархию ядра **/proc/device-tree**).

В зависимости от вашего дистрибутива и его политики безопасности, иногда частичный вывод Lshw доступен обычным пользователям, но читать DMI-таблицы разрешается только суперпользователю.

Lshw умеет создавать отчет в текстовом, XML или HTML форматах. Чтобы получить вывод в XML, укажите ключ **-xml**; для получения HTML-вывода используйте **-html**. Другие полезные опции командной строки позволяют управлять отчетом, генерируемым *Lshw*: например, указав опцию **-class display**, получить вывод списка только видеокарт, подключенных к вашей системе.

Графическая оболочка создана в *GTK* и доступна в виде дополнительной опции пакета *lshw*. Она позволяет вам просматривать иерархию устройств, составляющих вашу систему, начиная от материнской платы в качестве корня древа, и предоставляет ту же информацию, что и инструмент командной строки, вместе с графическими иконками, отображающими класс устройств. Так что не беритесь за отвертку, когда вам необходима некая информация о вашем оборудовании – на это есть *Hardware Lister*.

> «Hardware Lister может создать отчет как текст, XML или HTML.»



› Вывод показан в *GTK*-интерфейсе. Кстати, автор, Лайонел Винсент [Lyonel Vincent], использует уникальную систему нумерации версий. 'B' означает второе поколение *Lshw*.

Менеджер фотографий

FLPhoto

Версия 1.3 **Сайт** www.easysw.com/~mike/flphoto

Использование мощного *Gimp* для простого редактирования ваших фотоснимков – это стрельба из пушек по воробьям. Опять же, *Gimp* не имеет функции составления каталогов, а это становится серьезной проблемой, когда ваш жесткий диск забит сотнями фотографий.

FLPhoto решает все ваши проблемы управления фотографиями, включая загрузку изображений с камеры, редактирование и простую обработку, каталогизирование и печать. Его автор, Майкл Свит [Michael Sweet] – владелец Easy Software Products, фирмы, стоящей за CUPS – наиболее распространенной инфраструктуры печати в Linux.

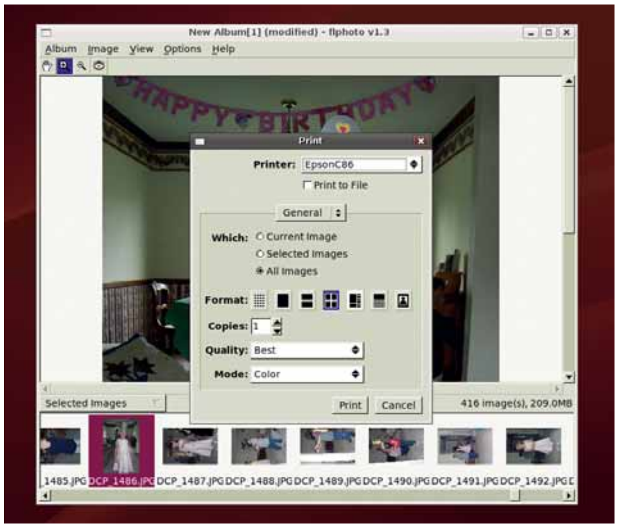
Префикс 'FL' в заголовке программы отражает тот факт, что *FLPhoto* построен на базе FLTK (Fast, Light Toolkit) – современного GUI-инструментария, компактного и крайне быстрого. *FLPhoto* это очень даже пошло на пользу, и он всегда работает быстро, даже оперируя альбомами из сотен изображений. Программа проста в использовании и укомплектована встроенной документацией.

FLPhoto стремится быть полезным на каждом этапе жизненного цикла цифровых изображений. С его помощью вы можете загружать кадры с вашей камеры (для связи с камерами используется стандартная библиотека *libgphoto*, поддерживающая сотни устройств), импортировать картинки с диска (и целыми каталогами, и отдельными файлами), аранжировать и манипулировать фото, экспортировать их в web-страницу или в слайд-шоу. Чудесно!

Структурной единицей *FLPhoto* является альбом – коллекция импортированных фотографий. Можно сортировать содержимое альбома по имени или дате, а изображения снабжать примечаниями. Фотографии при импорте не копируются: фактически, альбом – это коллекция ссылок на место физического расположения картинок на диске.

Картинная красота

Особенно впечатляет набор инструментов печати *FLPhoto*; что и не удивительно, с учетом бизнеса его автора. CUPS, естественно, поддерживается, и можно выбрать печать одного, двух или четырех изображений на странице или печать миниатюр. Есть и более изощренные форматы – печать календарей или матовых отпечатков, с выбором цветного шаблона определяемой пользователем ширины, который будет каймой отпечатка (превосходно для фотографий в рамках).



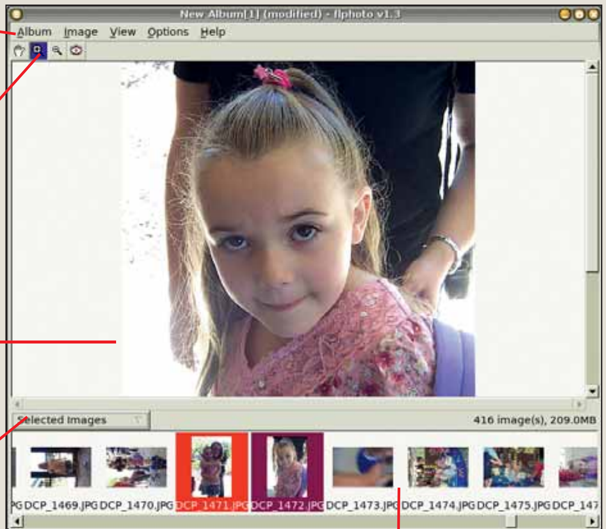
› В *FLPhoto* есть интересные опции печати, включая печать календарей с вашими фото – отличный подарок для родственников.

Как уже сказано, *FLPhoto* выполняет кое-какую обработку изображений. Задуман он не для конкуренции с полнофункциональными графическими пакетами, но обеспечивает большую часть необходимых операций, то есть вращение и обрезку изображения и несложные варианты ретуширования, вроде размывки (blur), наведения резкости (sharpen) и удаления красных глаз. Кроме последней, все эти операции могут применяться как к отдельным изображениям, так и в пакетном режиме ко всем помеченным картинкам альбома. Это здорово ускоряет работу, когда, например, у вас куча изображений с одинаковым дефектом.

И это не все. Наши и без того теплые чувства к программе стали совсем горячими от работы с невероятно удобным инструментом обрезки. Это не беззаботное «потяни рамку, авось лучше станет». Границу можно обрезать с точностью до пикселя и даже назначить ограничения на пропорции: выбрать или пропорции исходного фото, или один из стандартных форматов, например, 3:4. Это невероятно удобно, если вы хотите печатать на бумаге конкретного размера.

Как и все приложения, *FLPhoto* не идеален. Мы нашли странным, что при всей функциональной продвинутости отсутствуют горячие клавиши – особенно для навигации по альбому. Хотелось бы также видеть больше кнопок на панели инструментов для ускорения часто используемых операций. Тем не менее, *FLPhoto* – прекрасный пакет, и вполне заслуживает звания нашего выбора, Hottest Pick.

Исследуем интерфейс FLPhoto



Навигация
Используйте это меню для доступа к основным функциям *FLPhoto*, например, инструментам управления альбомом и редактированию выбранного изображения.

Важные инструменты
Панель инструментов позволит вам быстро войти в режимы панорамы, увеличения или удаления эффекта красных глаз для текущего изображения.

Ваш холст
Здесь отображается отдельная картинка. Она может быть увеличена, панорамирована, развернута, обрезана или обработана при помощи функций, например, фильтра резкости.

Групповое редактирование
Это выпадающее меню позволяет вам применять функции обработки изображений к группе выбранных картинок в альбоме.

Альбом
Картинки в текущем альбоме показаны в виде горизонтального списка, который может быть отсортирован по имени или дате в порядке возрастания или убывания.

HotGames Развлекательные приложения

Аркада

Frozen Bubble

LINUX FORMAT HotPicks повторный визит

Версия 2.1.0 **Сайт** www.frozen-bubble.org

Разработчики игр с открытым кодом должны быть благодарны Линусу Торвальдсу за решение создать свободное ядро для операционной системы GNU. Прикиньте, как сейчас назывались бы игры, если бы Тукс, симпатичный символ Linux, не явился на свет? Гонщик Гну? Брр! Одна из игр, удачно использующих всеобщего любимца – *Frozen Bubble*, реинкарнация классической аркады Puzzle Bobble, описанной в LXF43.

Если вы еще не встречались с подобными играми, то *Frozen Bubble* – это Tetris вверх ногами. Каждый уровень содержит массу цветных шаров в верху экрана. Наш герой Тукс стреляет в них шариками случайного цвета, и они прилипают ко всем шарам, которых коснутся, или к потолку; в процессе возможны рикошеты от боковых стен. Когда собирается группа из трех или более шаров одного цвета, они взрываются. Задача – очистить экран, взорвав таким способом все шары.

Frozen Bubble – отлично сделанная версия игры этого типа. Здесь профессиональны и графика, и звук; качество просто бьет в глаза. В версии 2.0 представлена сетевая игра по локальной сети или в Интернете (в зависимости от типа сервера) с поддержкой до пяти игроков. Естественно, поддерживаются одно- и двухпользовательские режимы на локальной машине, и добавлен новый режим одного игрока – «тренировка», для оттачивания мастерства в многопользовательской игре. Пакет дополнен редактором уровней.



› В режиме одного игрока *Frozen Bubble* – это 100 уровней удовольствия от взрывания пузырей для тех, у кого не густо с друзьями..

> «В режиме тренировки игрок-одиночка может оттачивать мастерство.»

Согласно нынешнему стандарту для игр с открытым кодом, *Frozen Bubble* создана при помощи Perl и SDL. Кроме привязок Perl-SDL, вам также понадобятся библиотеки *SDL_image*, *SDL_mixer* и *SDL_Pango*. Разработчики предоставляют для загрузки RPM для Mandriva, но многие другие дистрибутивы также включают пакет *Frozen Bubble*.

Головоломка

Enigma

Версия 1.00b **Сайт** www.nongnu.org/enigma

Эта головоломка отдает дань двум классикам прошлого: *Oxyd* для Atari ST и *Rock'n'Roll* для Amiga. Мы не играли в оригиналы, и не можем сказать, насколько точно это сравнение; однако *Enigma* более чем сильна, чтобы судить ее уже по собственным заслугам.

Игровой процесс *Enigma* обманчиво прост. Каждый уровень содержит скрытые пары цветных так называемых Oxyd-камней, и вы должны их открыть, дотронувшись до них вашим камушком, который вы передвигаете мышью.

Как всегда, жизнь непроста, и на вашем пути попадаются различные препятствия, включая лабиринты, двери, лучи лазера и прочее. Одни двери открываются ключами, другие – нажатием на плиты, с которых надо сперва отодвинуть камни, а третьи – переключателями. Луч лазера при прикосновении убивает, но его можно отразить зеркалом, и если он ударит в Oxyd-камень, то камень раскроется.

Это лишь немногие из разнообразия элементов, представленных в *Enigma*, но некоторые уровни проверят и ваше проворство в обращении с числами. Тип поверхности определяет, с какой скоростью катится ваш камушек и насколько трудно им управлять. Некоторые поверхности наклонны, и вам необходимо развить значительную скорость, чтобы взобраться на них. Другие поверхности, например, водные, вообще нельзя пересечь. К счастью, в *Enigma* есть серия обучающих уровней, где вы знакомитесь с типами будущих препятствий.



› Перегруженный лазерами уровень Enigma уместно назван «Миссия невыполнима». Эх, сюда бы Тома Круза...

> «Игровой процесс, уже и так захватывающий, к счастью, сохранен.»

В последний раз мы рассматривали *Enigma* в LXF49, три года назад. С тех пор игра стала более красивой и лощеной, но игровой процесс, уже и так чертовски захватывающий, сохранился. Свежая *Enigma* теперь упакована 750-ю уровнями, чтобы удивить и поразить вас, и вы можете сравнить свое мастерство с собратьями Enigmatic'ами по всему миру, благодаря наличию функции загрузки мировых рекордов времени прохождения уровней. Кому нравятся камушки в *Elgin* или сферы в *Sofia*, хватайте свои шары и вперед!

Помощь в запоминании

JMemorize

Версия 1.0.0 **Сайт** http://jmemorize.org

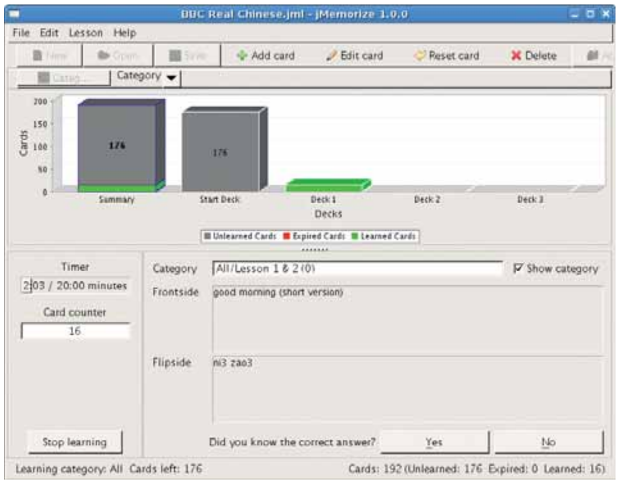
Даже в нашем пронизанном коммуникациями мире, где Google и Wikipedia доступны в два щелчка, учителя все еще настаивают на зубрежке. Возмутительно! Что ж, если вы из студентов, неспособных запомнить что-либо, не отображающееся на экране компьютера, *JMemorize* может помочь вам в подготовке к очередному экзамену.

JMemorize – Java-приложение, использующее популярный способ запоминания: всплывающие карточки. Ну, вы знаете: на одной стороне карточки – вопрос, например, «Какова скорость полета ласточки?», а на другой – ответ. Вы создаете набор карточек, содержащих все факты, которые необходимо запомнить. Чтобы проверить себя, берете карточку, задаете себе вопрос, а затем переворачиваете ее и сравниваете свой ответ с верным.

JMemorize написано на Java и должно запускаться на любой системе, поддерживаемой средой Java. В последней GNU-среде Java, *gij 4.1*, мы потерпели неудачу – видимо, потому, что ее реализация Free Swing работает еще не совсем верно. Так что для запуска *JMemorize* вам потребуется приличная вещь: Sun JVM. Для запуска программы просто введите **java -jar JMemorize-1.0.0.jar**.

Запустившись, *JMemorize* предоставит средства для создания, манипулирования и хранения набора виртуальных карточек. В режиме просмотра *JMemorize* использует систему Лейтнера [Leitner], облегчающую запоминание: повторение с интервалами. Возможно, вы ее знаете. Карточки сортируются на «изученные» и «не изученные». *JMemorize* выводит вам карточку из стопки «не изучено», а затем показывает ответ. А вы должны указать, дали ли вы верный ответ, щелкнув на Да или Нет. Карточки с которыми вы не справились, идут обратно в группу «не изучено», а если ответ верен, то в группу «изучено». Поэтому оно заставляет вас повторять вопрос, который вы не можете удержать в памяти.

> «Применяет популярный способ запоминания – всплывающие карточки.»



› Используя Swing, *JMemorize* не ласкает глаз, зато и не отвлекает от обучения лишними деталями.

JMemorize дополняет эту простую схему. Например, можно установить временной лимит для каждой карточки и для всей сессии, предоставляются инструменты для импорта и экспорта карточек в виде CSV-файлов и экспорт в PDF- или RTF-формат. Репетиторам действительно нет оправдания...

Системный монитор

Atop

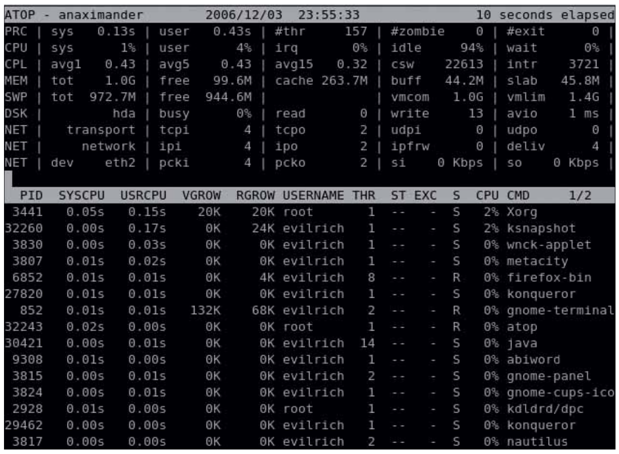
Версия 1.17 **Сайт** www.atconsultancy.nl/atopx

Мы в *Linux Format* не адепты графического интерфейса, и в доказательство этого стремимся по возможности представлять в HotPicks чисто текстовые приложения (к большому огорчению нашего художественного редактора). Для системного монитора *Atop* тот факт, что оно консольное, является преимуществом. Ведь это инструмент сбора различной системной статистики, поэтому обязан минимально влиять на систему: кому нужно, чтобы он вмешивался в измеряемые данные?

Но постойте. Зачем вообще нужен еще один текстовый системный монитор? Разве мало стандартной команды *top*? Ответ в том, что *Atop* предоставляет намного больше данных, чем простой старый *top*, и с ним легче работать. Кроме обычных величин использования памяти и процессорного времени, *Atop* может составлять профили сетевой и дисковой активности для каждого процесса. Правда, для этого он требует нескольких заплаток ядра: без них *Atop* сообщает только об активности сети и диска во всей системе. На домашней странице *Atop* доступны заплатки для ядер 2.6 (в настоящий момент вплоть до 2.6.18.3). Инструкции по применению этих заплаток прилагаются, но вы, конечно, и сами знаете, как собирать и настраивать ядро – это вне рамок обзора HotPicks, но хорошо описано Нейлом Ботвиком на стр. 70.

Экран *Atop* разбит на две части. Верхняя половина отображает различную системную статистику за последний интервал (по умолчанию – 10 секунд, но его можно изменить через параметры командной строки или нажав клавишу i при запущенном *Atop*). Сюда входит использование процессора, нагрузка, потребление памяти или дисковая и сетевая активность. В нижней половине *Atop* отображает информацию по отдельным процессам, по умолчанию – использование процессора, занимаемую память, число потоков и статус.

> «Atop составляет профили дисковой и сетевой активности для каждого процесса.»



› **Системный монитор типа *Atop* засекает, какой процесс сжирает все время процессора, память и ширину канала.**

Нажмите M для получения более детальной информации об использовании памяти, S – о параметрах планировщика (включая политику, значение nice и приоритет), D – об использовании диска (если вы применили описанные выше заплатки), и так далее. По умолчанию *Atop* отображает только процессы, активные в последнем интервале, но это можно переключить нажатием клавиши A. Можно также сортировать список процессов по различным критериям, и если надо узнать, кто сожрал все процессорное время, вы знаете, к чему обратиться.

Последовательный терминал

CuteCom

Версия 0.14.1 **Сайт** http://cutecom.sourceforge.net

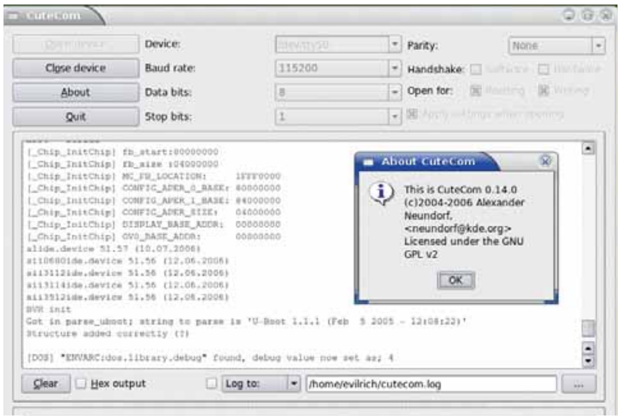
Несмотря на почти повсеместную замену на более современные технологии, скромный последовательный порт все еще приносит пользу. Мало кто из нас сегодня входит на сервер при помощи терминала или получает доступ к доскам объявлений через последовательный модем, но простота последовательных соединений все еще делает их идеальными для отладки, особенно для отладки низкоуровневого кода.

Если вам почему-либо надо эмулировать последовательный терминал на рабочем столе, *CuteCom* – прекрасный способ это сделать. Он сравним с классическим Minicom, но это не текстовое приложение, ибо создано при помощи инструментария *Qt*. Потому *CuteCom* прост в использовании, и все его функции доступны по кнопкам, а не через легко забываемые клавиши.

Можно запросто настроить параметры соединения, выводить результаты в текстовом или шестнадцатеричном формате, и поддерживаются различные режимы линии связи – обычные последовательные протоколы для передачи файлов и ведение журнала. *CuteCom* отображает полученный и введенный текст в разных окнах, так что их легко отличить, и имеет историю ввода.

Сборка *CuteCom* – сама простота, если у вас установлены инструменты разработчика *Qt*. Просто вызовите **configure** (обертку для *Qt*'шного qmake) и **make**. После всех этих скучных текстовиков, насладитесь светлым графическим пятном последовательного терминала.

› Пусть это продукт шестидесятых, но старый добрый последовательный порт всегда в моде.



Дизассемблер

Dissy

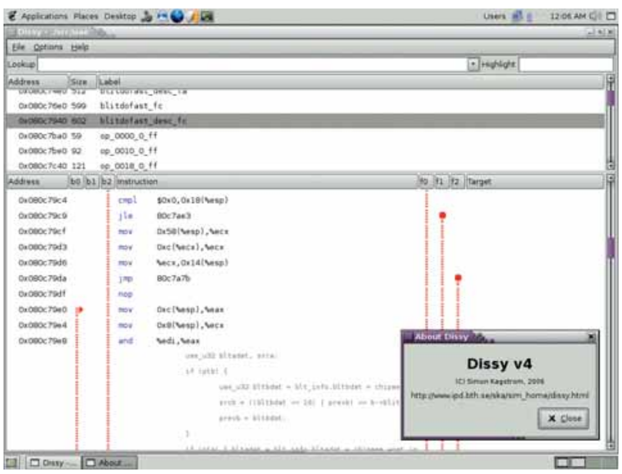
Версия 4 **Сайт** http://rtlab.tekproj.bth.se/wiki/index.php/Dissy

Технология компиляторов далеко шагнула за последние годы, но все равно бывает нужно проверить, что же выдал компилятор: или подозревая, что он неверно понял ваш код, или думая, что кое-что вы сами могли бы сделать лучше. В таких случаях необходимо умение дизассемблировать объектный код.

Инструмент *objdump* из пакета **binutils** от GNU может дизассемблировать объектный код в обычный текст на языке ассемблера, но с его выводом не очень-то поработаешь. Тут и появляется *Dissy* – графическая оболочка для binutils, созданная при помощи Python и *GTK*.

Двухпанельное окно *Dissy* показывает список меток наверху и дизассемблированное внизу. Щелчок на метке показывает дизассемблированный объектный код около нее. Если вы скомпилировали код с поддержкой отладки, рядом с дизассемблированным объектным кодом *Dissy* покажет исходный код. На некоторых поддерживаемых архитектурах (сейчас это x86, PowerPC и MIPS) *Dissy* может также подсвечивать ветки команд ветвления, помещая красную стрелку от источника до назначения – щелкните мышью по инструкции для перехода между ними. Если вы вывихнули глаза, разбирая вывод *objdump*, попробуйте *Dissy*. LXF

› *Dissy* добавляет к полученному коду аннотации, способствующие пониманию.



Также вышли

Новые и обновленные приложения, также заслуживающие внимания

» **AfterStep 2.2.4** Оконный менеджер в духе Nextstep. http://www.afterstep.org

» **Bakefile 0.2.1** Генератор make-файлов. http://bakefile.sourceforge.net

» **Ecksdee 0.0.9** Игра-гонка на основе движка Crystal Space 3D. http://ecksdee.sourceforge.net



› *Ecksdee* – футуристические гонки.

» **Fox Desktop 0.1.12** Легковесное окружение рабочего стола. http://fifthplanet.net

» **Gag 4.7** Графический менеджер загрузки. http://gag.sourceforge.net

» **GPG-Crypter 0.2.1-1** GTK-оболочка для GnuPG. http://gpg-crypter.sourceforge.net

» **Gwenview 1.4.1** Просмотрщик изображений для KDE. http://gwenview.sourceforge.net



› Просмотр изображений с *Gwenview*.

» **GPutty 0.9.9** Клон Putty для рабочего стола Gnome. http://people.defora.org/~khorben/projects/gputty

» **Jackbeat 0.6** Audio-секвенсер. www.samalyse.com/jackbeat

» **KeyTouch 3.1.0** Настраивает мультимедиа-клавиши на клавиатуре. http://keytouch.sourceforge.net

» **Laurux 1.07** Приложение для бухучета и выставления счетов. www.laurux.fr

» **Pinot 0.63** Инструмент поиска и индексирования документов. http://pinot.berlios.de

» **UMLet 7.1** Простой инструмент для рисования UML-диаграмм. www.umlet.com

» **Warrior 0.96** Чистый Java web-браузер. http://html.xamjwg.org

» **Xournal 0.3.2** Записная книжка и альбом для Tablet-PC. http://xournal.sourceforge.net

LXF DVD89

Хакеры, на старт: встречайте новый релиз OpenSUSE!



Майк Сондерс любовно подбирает содержимое диска *Linux Format*, а также поддерживает сайт www.linuxformat.co.uk.

И больше, и лучше

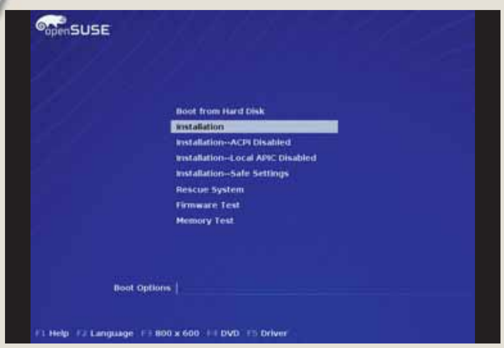
Перед тем, как нырнуть в обзоры авангардного ПО с DVD этого месяца, прослушайте пару объявлений. Во-первых, для упрощения работы с диском, мы увеличили наш раздел на две страницы, так что не переживайте, что под этим вступлением нет содержания диска – оно теперь на стр. 123. Больший объем означает большее внимание к пошаговой установке – будем описывать ее подробнее (что особенно ценно, если вы еще никогда не устанавливали тот или иной дистрибутив), и большее внимание к программам, до этого упоминаемым лишь вскользь.

Ну, а теперь – о самом DVD: в течение последней декады SUSE был одним из популярнейших дистрибутивов, и мы рады предложить вам новый релиз сообщества 10.2, полный обновлениями пакетов и снабженный новым меню KDE и усовершенствованной системой онлайн-обновлений. Мы включили также несколько статей в формате PDF из предыдущих номеров Linux Format, в том числе два спецрепортажа и учебник по работе с *OpenOffice.org*, плюс набор приложений Mono, новые инструменты рабочего стола и многое другое. Как говорится в знаменитом загрузочном сообщении SUSE: «Наслаждайтесь!»

mike.saunders@futurenet.co.uk

ВНИМАНИЕ! Обложку диска можно скачать на нашем сайте http://linuxformat.ru

Шаг за шагом: Установка OpenSUSE 10.2



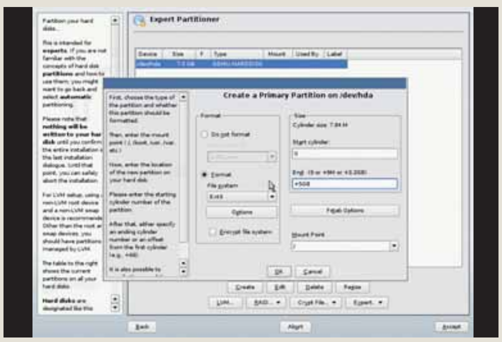
1 Загрузка

Загрузите компьютер с LXF DVD. Выберите Installation; если появятся проблемы – перезагрузитесь и выберите Safe Settings.



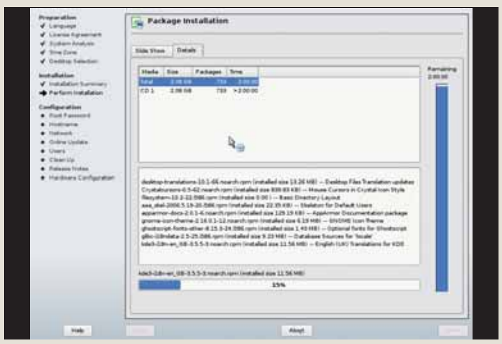
2 Сведения

Во время загрузки OpenSUSE можно нажать Esc, чтобы отключить графический режим и видеть сообщения загрузчика. Это не обязательно, но может пригодиться, если процесс вдруг остановится.



5 Разбиение

Выберите Create Custom Partition Setup, затем – Custom Partitioning. Отведите на корневой раздел ext3 (/) 5 ГБ или больше для основной файловой системы Linux плюс 512 МБ на раздел подкачки.



6 Загрузка

Программа установки OpenSUSE начнет копировать файлы. Это может занять около часа. Откройте вкладку Details, чтобы видеть состояние установки.

Дистрибутив Linux

OpenSUSE 10.2

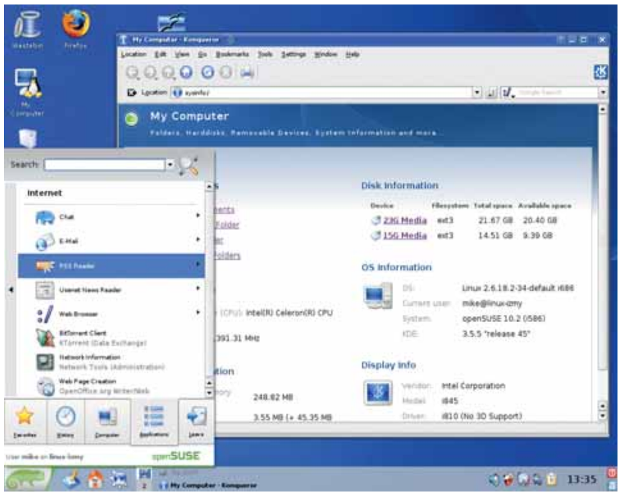
При всем брюзжании, вызванным сделкой Novell с Microsoft, нельзя отрицать, что OpenSUSE – превосходная разработка, что и подтверждается стабильностью его второго места в рейтинге Distrowatch Hit List.

OpenSUSE имеет более чем десятилетнюю историю, пользуется корпоративной поддержкой Novell, а теперь привлекает энергичное сообщество – ради создания превосходного и всеобъемлющего дистрибутива для настольных ПК, серверов и рабочих станций. Все это – отличная рекомендация нового релиза 10.2 (32-bit x86) на нашем DVD, готового к установке и работе. Он включает обновления KDE, Gnome, *Firefox* и многих других приложений, плюс новое затейливое меню KDE и усовершенствованную систему онлайн-обновлений. Чтобы установить его, загрузите компьютер с нашего DVD. Чтобы установить OpenSUSE 10.2 на машину, где имеется только CD-ROM, используйте систему Jidgo – см. **index.html** на DVD.



› В новом меню KDE, присмотритесь к переключаемым панелям внизу.

Необходимый минимум для работы OpenSUSE 10.2 – 256 Мб ОЗУ и процессор 500 МГц, но с такими данными вам лучше устанавливать не KDE или Gnome, а выбрать более легковесный рабочий стол – например, Xfce, *FVWM* или *Window Maker*. Для обеспечения достойной работы рабочего стола по умолчанию требуется ПК с процессором 1 ГГц и выше и не менее 512 Мб ОЗУ. По части жесткого диска, рекомендуем отвести не менее 5 Гб под раздел OpenSUSE. Внизу слева вы найдете описание процесса установки, а через страницу – полезные советы и подсказки по настройке. Если во время установки или работы с OpenSUSE возникнут проблемы, обращайтесь на сайт проекта – www.opensuse.org. Посетите также форумы SUSE на www.linuxforum.ru, где другие пользователи OpenSUSE, возможно, помогут вам справиться с вашими проблемами.



Установив OpenSUSE, приступим к знакомству с рабочим столом. Если вы – постоянный пользователь Linux, то вы уже знакомы с настройками по умолчанию рабочего сто- »

› Под значком My Computer в OpenSUSE таится масса информации о разделах жесткого диска, папках и об оборудовании.



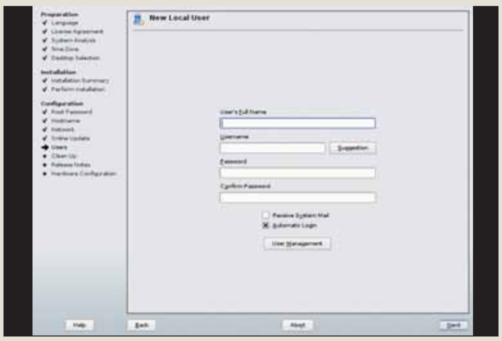
3 Запуск

Выберите язык и прочтите лицензионное соглашение. Выберите Upgrade, если у вас уже стоит SUSE – в противном случае, выберите New Installation. На экране выбора рабочего стола выберите KDE, если не уверены, что взять.



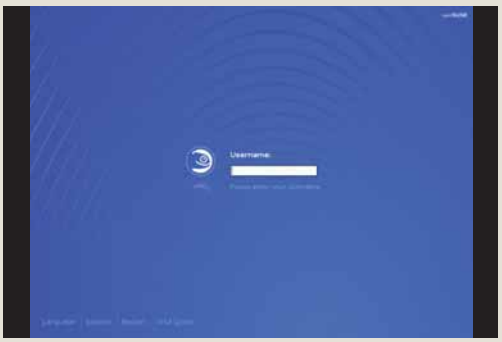
4 Отладка

Примите настройки конфигурации по умолчанию, если они вас устраивают, и переходите к шагу 6. Если хотите изменить разделы жесткого диска, нажмите Partitioning и переходите к следующему шагу.



7 Перезапуск

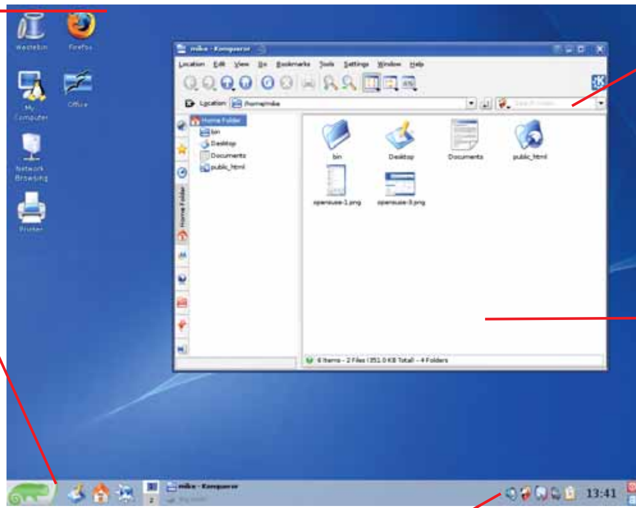
После всего этого система будет перезагружена для дополнительной настройки. Установите пароли для root (администратора) и пользователей. Вы можете спокойно прервать ZenWorks, если вам покажется, что он тормозит.



8 Вход в систему

После настройки ваших сетевой и видеокарты программа установки завершит работу, и вы можете войти в систему, используя новое имя и пароль.

Изучаем рабочий стол OpenSUSE



Значки
Значки на рабочем столе обеспечивают быстрый доступ к программе просмотра файлов, *Firefox* и *OpenOffice.org*. При желании можно добавить сюда значки других программ и документов.

Главное меню
Нажмите на Geeko, значок с гекконом-талисманом SUSE – откроется меню запуска основных программ, из него можно запускать программы и получать доступ к настройкам.

Системная панель
Здесь показываются значки многих работающих программ.

Beagle
Файловый менеджер интегрирован с *Beagle*, настольной системой поиска по содержимому документов.

Konqueror
Konqueror, файловый менеджер, может просматривать удаленные сервера через SMB и SSH. Может также достойно служить web-браузером.

» ла KDE (или Gnome, если вы выбрали его во время установки), но имеется одно заметное дополнение, достойное рассмотрения: новое меню KDE. Команда разработки OpenSUSE отказалась от стандартного – и слегка перенасыщенного – меню KDE в пользу панельной альтернативы, позволяющей переключаться на разные варианты отображения.

Звучит не очень понятно? Ну, у вас есть вид Favorites, где показаны часто используемые программы, потом History, там содержится список программ и документов, с которыми вы работали недавно. Вид Computer дает доступ к разделам на жестком диске и центру управления *Yast*, а Applications предоставляет древовидный перечень установленных программ. Какое-то время, конечно, уйдет на привыкание, но зато потом это меню становится вашей второй натурой, а строка поиска вверху этого меню – еще один прелестный штришок.

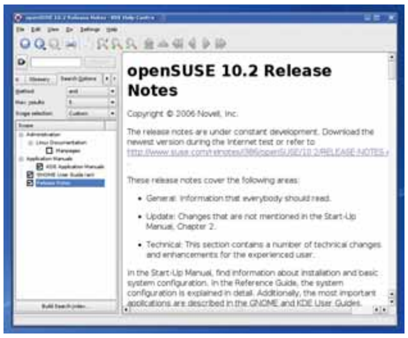
Настройка

Первым делом после установки дистрибутива хочется настроить его в соответствии со своими предпочтениями – добавить какие-то программы, изменить разрешение экрана, и т.д., и т.п. Большинство задач администрирования решает всеобъемлющий инструмент *Yast*, это наиболее известная функция OpenSUSE и весьма зрелая программа настройки. Для запуска *Yast*, нажмите Main menu > Computer > Yast (надпись "Administrator Settings" над значком). Вас попросят ввести пароль администратора, который вы задали при установке.

Когда появится основное окно *Yast*, вы увидите, что имеется выбор между разными типами администрирования – можно настраивать программы, оборудование, сеть, безопасность и многое другое. Для переключения между режимами, используйте панель слева, а затем на значок на правой панели, чтобы выбрать желаемое действие. Например, чтобы изменить разрешение экрана, нажмите на Hardware слева, затем – Graphics Card And Monitor справа. Если вы отведете время на ознакомление со всеми опциями, предлагаемыми *Yast*, это значительно сэкономит ваши усилия в будущем, когда вы решите что-то поменять!

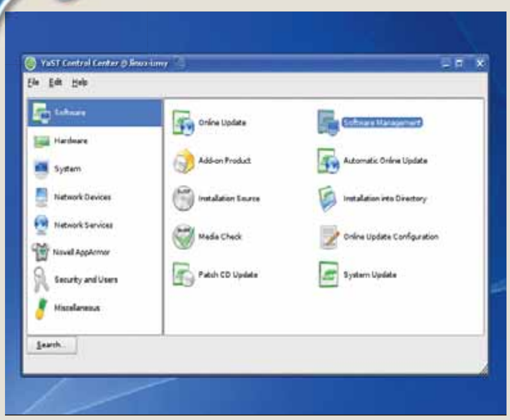
По части косметики рабочего стола – ну, поменять шрифт, выбрать тему или украсить окно – вам понадобится Центр Управления KDE (KDE Control Centre), куда можно попасть через Главное меню > Favorites (значок Configure Desktop). Как и в *Yast*, откроется двухпанельное окно, с категориями настройки слева. Для выхода из OpenSUSE нажмите на Главное меню > Leave, а затем – на красную кнопку Shutdown.

В OpenSUSE 10.1 было несколько ошибок, связанных с онлайн-обновлением, но в релизе 10.2 они исправлены, и получить обновления исправления ошибок или безопасности можно всего лишь несколькими щелчками мышью. В *Yast* выберите панель Software, затем щелкните на Online Update, и из Интернета загрузится список самых новых пакетов. Наконец, ниже приведены краткие рекомендации по установке дополнительных программ с нашего DVD. Приятного знакомства с новым OpenSUSE!



› Выбрав в главном меню Favorites > Help, вы увидите обзор релиза и справку о KDE.

Шаг за шагом: Установка программ с помощью Yast



1 Запуск
Откройте главное меню, затем – Computer > YaST, и выберите Software на левой панели и Software Management на правой. Возможно, вам придется ввести пароль администратора.



2 Выбор
Чтобы найти необходимый пакет, введите критерий поиск ана левой панели или смените тип фильтра (сверху) на Patterns, чтобы просмотреть группы пакетов.

Дистрибутив Linux

Damn Small Linux 3.1

Нам в редакции LXF чаще всего задают вопрос: «Почему Linux не идет на старом «железе»?» Многие новички, ознакомившись с аппетитами крупных новых дистрибутивов с мощными средами рабочего стола, типа SUSE или Fedora Core, находят странным постоянное утверждение сообщества Linux, что требования Microsoft раздуты. Если для Fedora действительно подавай 512 МБ ОЗУ для приличной работы, то чем она лучше Windows XP? Где же пресловутая «легковесная альтернатива», о которой так много говорили?

Ну, как и со многим другим в Линуксландии, это вопрос выбора дистрибутива. Fedora, SUSE и Ubuntu идут на все ради эргономичности, а солидные рабочие столы стараются удовлетворить запросы пользователей современных ПК – вот и приходится жертвовать поддержкой старых машин.

Damn Small Linux (DSL) исповедует противоположный подход: он избегает ресурсоемких приложений вроде KDE, Gnome и *OpenOffice.org*, заменяя их нетребовательными к ресурсам (но удобными и полезными) эквивалентами, например, *IceWM*, *Fluxbox* и Ted. И правда, все приложения, включенные в ISO-образ весом 50 МБ, требуют минимума памяти, главные примеры тому – почтовый клиент *Sylpheed*, web-браузер *Dillo* и клиент обмена мгновенными сообщениями *Naim*. Пинками можно заставить Damn Small Linux 3.1 работать и на 486 машине с 16 МБ ОЗУ, но для приемлемой производительности все же рекомендуем 32 МБ ОЗУ.



› Если *Fluxbox* вас не потрясает, берите Windows-подобный рабочий стол *IceWM*.

У многих из нас зря пылится один или несколько старых ПК, а Damn Small Linux может оживить эти машины, установив на них полезные программы. От читателей Linux Format поступают очень позитивные отзывы о DSL – один из них умудрился воскресить с помощью этого дистрибутива позабытый старый Pentium и отдал его своим детям – пускай играют! Учитывая, сколько ПК выбрасывается (в частности, предприятиями) по той причине, что «их время прошло» – здорово, что Linux предлагает способ защитить окружающую среду от компьютеров.

Для установки Damn Small Linux 3.1 вам потребуется записать один из ISO-образов DSL на CD-R. Эти ISO – образы CD, поэтому вы не сможете просто скопировать их на диск, как обычные файлы; найдите в вашей программе записи CD опцию записи ISO-образов (Burn ISO Image). Например, в *K3b* вам надо зайти в Tools > Burn CD Image. В командной строке используйте утилиту *cdrecord* таким образом:

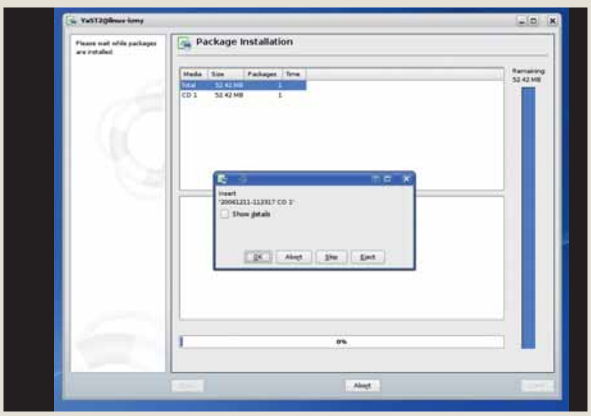
```
cdrecord -v -dao speed=24 dev=/dev/cdrom
dsl-3.1.iso
```

Запишите диск и загрузите ваш ПК с образа **dsl-3.1.iso** (в разделе DVD **Distros/DamnSmallLinux**), а если не получится, попробуйте версию **dsl-3.1-syslinux.iso**. Можете попробовать дистрибутив в *VMware* с помощью **dsl-3.1-vmx.zip**.

› Обзор Damn Small Linux см. на стр. 11.



› Прямо «шведский стол»… Damn Small Linux втиснул в свои 50 МБ бездну программ.



3 Установка

Выберите на правой панели программы, которые вы хотите установить, нажмите Accept (справа внизу) и после подсказки вставьте LXF DVD.

Документация

PDF-файлы журнала

Как и на прошлых трех дисках, на DVD этого месяца – не только программы: мы включили в него 42 страницы спецрепортажей и учебников из предыдущих номеров *Linux Format*. Вы можете просто прочитать их из вашего браузера в формате PDF – откройте **index.html** на диске, найдите раздел **Magazine/PDFs** – и расширяйте ваши знания о технологиях Linux! Даже если вы – постоянный читатель, вы можете собрать PDF файлы и создать библиотеку документации Linux (чтение с помощью PDA где-нибудь в поезде особо полезно...).

Мы сбросили сюда материал LXF63 Все о Сети (Network Everything) – 16-страничное мега-руководство по сетям в Linux. Хотите настроить Bluetooth? Мы расскажем, как. Хотите работать с удаленным рабочим столом? Все это вы найдете здесь. Мечтаете испробовать свои силы в распределенной компиляции? Да, и этот вопрос глубоко затронут, так же, как и SSH, Wi-Fi, безопасность и многое другое.

Следующая большая подборка – с меньшим техническим уклоном: статья LXF80 о Libre Graphics, которая рассматривает чарующий мир графического ПО Linux. Принято считать, что Mac – платформа де-факто для дизайна и работы с изображениями и схемами, однако Linux тоже способен творить потрясающие вещи в этих областях, что и подтвердила конференция Libre Graphics, прошедшая прошлой весной в Лионе. Мы анализируем состояние графического ПО Linux, рассказываем о последних разработках и беседуем с кодерами *Scribus*, *Inkscape* и *Gimp* о том, что у них в заначке.



› Проблемы с оборудованием? См. наше 16-страничное руководство по сетям.

И, наконец, здесь есть четыре учебника по *OpenOffice.org*. Если вы только что перешли на Linux, или ищете информацию об эквивалентах *MS Office* с открытым кодом, прочтите их, дабы получить максимальную пользу. Они рассматривают работу с текстом, презентациями и базами данных.

»

Рабочий стол

Мегапак Mono!

Мы в редакции LXF любим Mono. Пусть это и реализация .NET, технологии Microsoft, но на своем месте она незаменимы – .NET and C# помогают создавать превосходные программы. Возможно, вы уже прочли статью про Mono на стр. 22, и в дополнение мы предлагаем вам все, что может вам понадобиться при знакомстве, использовании и разработке, включая самые свежие релизы приложений на базе Mono. Несомненно, C# – замечательный язык, он позволяет разработчикам создавать многофункциональные программы типа F-Spot без особых забот, и мы ставим ему «отлично»!

Многие дистрибутивы стали включать Mono в свою установку по умолчанию, но если в вашем дистрибутиве ее нет (или вы работаете на старом релизе 1.1), получите версию 1.2 из раздела **Разработка** нашего DVD. Весьма кстати, что команда разработчиков Mono не стала связываться с пакетами, предоставив нам простенький двоичный инсталлятор: он может мигом установить очень много чего. Скопируйте **mono-1.2-installer.bin** с нашего DVD в домашнюю директорию, откройте терминал, переключитесь в режим суперпользователя (например, *su* или *sudo bash*) и введите

```
chmod +x mono-1.2-installer.bin ./mono-1.2-installer.bin
```

Первая строка делает файл установщика Mono исполняемым, а вторая запускает его. Появится мастер установки на базе *GTK* – по его инструкциям вы пройдете весь путь установки Mono на вашу систему. Как и при любом обновлении ПО, сначала следует удалить старые версии Mono, чтобы не вышло каких-нибудь жутких накладок.

После установки Mono у вас появляются библиотеки, привязки *GTK* и компилятор *gmcs* – то есть все, что нужно для работы и разработки программ .NET. Мы сейчас как раз публикуем серию руководств по C# – самому популярному языку программирования .NET – см. стр. 58. В нашем разделе **Разработка** на DVD вы также найдете *MonoDevelop*, молодую, но очень солидную IDE для .NET-кодеров. Попробуйте RPM-пакет, если вы работаете в SUSE, Fedora или Mandriva, а если не сработает, соберите ее из исходных текстов – они в архиве **monodevelop-0.12.tar.gz**.

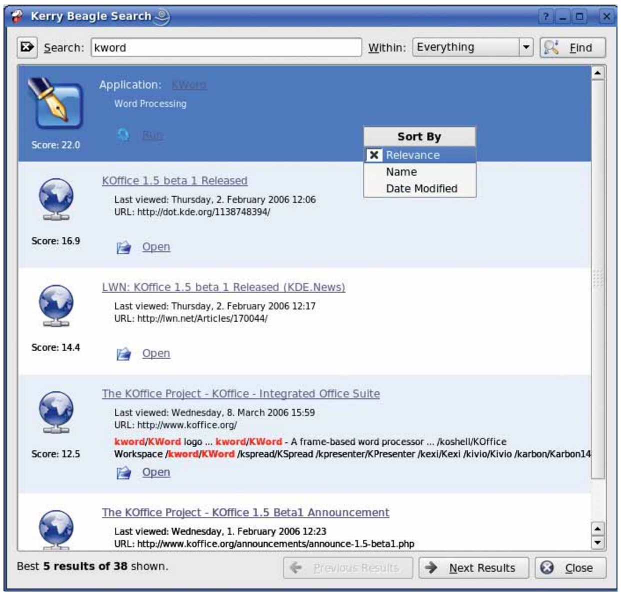
Звезды рабочего стола

Итак, Mono – это среда, а что там с приложениями? Наши разделы **Рабочий** стол и **Интернет** полнехоньки лучшими приложениями Mono, многим из которых посвящены статьи этого месяца. Во главе списка – *F-Spot*, выпустивший релиз 0.3.0 с новым руководством пользователя и исправивший более 50 ошибок. На DVD вы найдете полный исходный код и парочку труднонаходимых зависимостей.

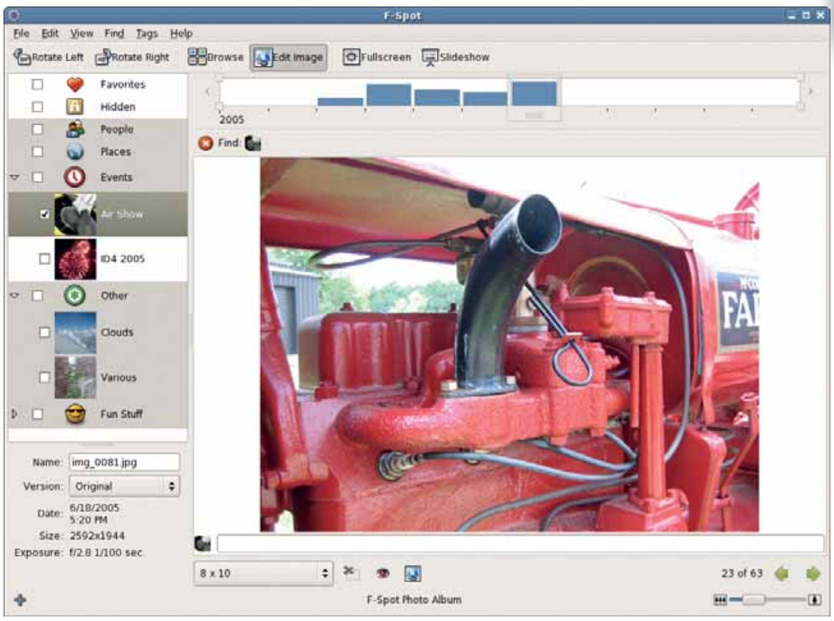
Еще одна прелестная программа – которая экономит массу времени, если вам нужно многое отслеживать – это *Tomboy*, wiki рабочего стола. Вместе с имеющимся стабильным релизом (0.4.1) мы включили самую свежую версию разработчиков (0.5.1) с улучшенным использованием D-BUS и переработанным поисковым интерфейсом. По части Интернет – если ваша читалка RSS-новостей как-то не радует, испробуйте *Blam*, его интерфейс ровен и отлично продуман.

И, наконец, хотя большинство программ Mono тяготеют к рабочему столу Gnome, пользователи KDE тоже не обойдены вниманием в .NET, благодаря *Kerry*, интерфейсу KDE для поисковой машины *Beagle*. *Kerry* глубоко интегрирован в KDE и помогает вам находить почтовые сообщения, документы, протоколы мгновенного обмена сообщениями и многие другие файлы по их содержимому, а не только по именам файлов.

В Интернет расцветает немало проектов на Mono, и если вы наткнетесь на приложения, достойные включения в наш DVD – обязательно сообщите нам, и мы их непременно включим!

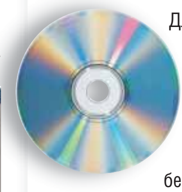


› У вас KDE? Все равно можно хлебнуть из фляжки Mono – с помощью *Kerry*, интерфейса KDE к *Beagle*.



› *F-Spot* – флагман среди приложений Mono, он легко управляется с фотографиями.

И наконец...



Для эффектного завершения этого пиршества программ – как насчет игр? Наш фаворит – очаровательно-нелепая *Cultivation*, где сообщество садовников совместно выращивает разные съедобные штуковины на выделенном участке. Игра описана как «социальный симулятор», в котором вы «знакомитесь с бесконечным в своем виртуальном разнообразии спектром различных растений и технологий садоводства». Да, непохоже на обычную бешеную стрелялку, где сначала лупят по кнопке, а уж потом думают! Поклонники игр в стиле ретро должны обратить внимание на *Magicor* – это версия классического *Ключа Соломона* (Solomon's Key) от Tecmo, дошедшая до нас с 8-битных дней (и вновь запущенного на Wii). Да, вместо Даны теперь ваш персонаж – пингвин, но это клише не портит отличный игровой процесс. LXF



› *Cultivation*: вот что мы именуем странным! С точки зрения садово-огороднических навыков.



Содержание DVD

Журнал
Список статей предыдущих выпусков LXF
Gtk..........Код из учебника GTK+
JavaEE..........Код адресной книги и Jetty 0.6.0rc3
Mono..........Код из учебника Mono.
PDFs..........Статьи предыдущих выпусков LXF.
Roundup..........Программы-трекеры.
Ruby..........Код из статьи.
Unix API..........Код примеров статьи.

Рабочий стол
Beagle..........Поисковый инструмент.
F-Spot..........Менеджер фотоколлекций.
Kerry..........Интерфейс KDE к Beagle.
PeaZip..........Распаковщик архивов.
Tomboy..........Программа для заметок.
Wyneken..........Текстовый редактор.
Xpdf..........Программа просмотра PDF.

Разработка
Medit..........Текстовый редактор.
Mono..........Открытая реализация .NET.
MonoDevelop..........C# IDE.
Nasm..........Ассемблер x86.
Blender..........Модель Tux'а
Ruby..........Язык программирования

Дистрибутивы
Damn Small Linux....Нетребовательный к ресурсам дистрибутив.
SUSE..........Дистрибутив, спонсируемый Novell.

Игры
Cultivation..........Социальный симулятор.
Magicor..........Игра-головоломка.
TuxWordSmith..........Обучающая игра.

Справка
Rute..........Руководство по администрированию Linux.

Hotpicks
Atop.......... Мониторинг процессов.
CuteCom.......... Последовательный терминал.
Dissy.......... Дизассемблер.
Enigma.......... Игра-головоломка.
FLPhoto.......... Программа управления изображениями.
Frozen Bubble.......... Клон Bust-a-Move.
Goggles.......... Интерфейс для DVD-плейера Ogle.
Hardware Lister.......... Сведения о вашем оборудовании.
JMemorize.......... Программа для зубрежки.
NSPluginWrapper.... Обертка для модулей Netscape.

Интернет
Blam..........Читалка новостей RSS.
Firefox extensions...Дополнения к Firefox.
Tor..........Анонимайзер.
Tork..........Контроллер Tor для KDE.

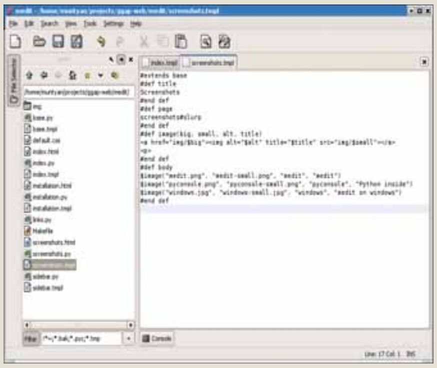
Безопасность
FloppyFW..........Маршрутизатор на дискете.
JPasswordGenerator..Генератор паролей.
Sussen..........Сканер безопасности.

Система
D-BUS..........Система пересылки сообщений.
GXInstall..........Инсталлятор программ.
Pinger..........Программа для ping'а нескольких узлов.
Qemu..........Эмулятор ПК.
Refit..........Загрузчик для Intel Mac.
Tilda..........Quake-подобный терминал.

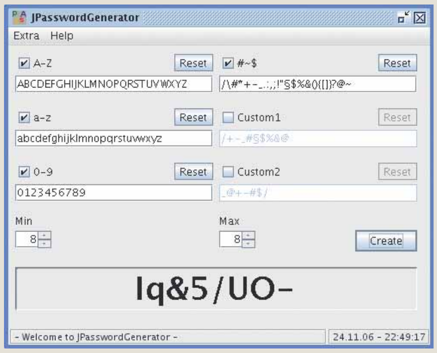
Сервер
Apache..........Web-сервер.
MyJSQLView..........Программа просмотра базы данных.
PostgreSQL..........Сервер баз данных.

Звук
AudioFormat..........Конвертор звуковых форматов.
Banshee..........Плейер и музыкальный менеджер.
Gnomeradio..........Тюнер FM-радио.
Gnormalize..........Аудиоконвертор.

Главное
Avifile..........Библиотека чтения/записи AVI-файлов.
Bash..........Командная оболочка.
CheckInstall..........Создатель бинарных пакетов.
Coreutils..........Утилиты командной строки.
CSV..........Индекс файлов диска.
GLib..........Низкоуровневая библиотека C.
Glibc..........Библиотека GNU C.
GTK..........Инструментарий пользовательского интерфейса.
HardInfo..........Информация о системе и тесты.
Jigdo..........Создатель ISO-образов.
Kernel..........Свежий релиз ядра Linux.
libsigc..........Система обратных вызовов для C++.
libXML..........Анализатор и инструментарий XML.
ncurses..........Оконный инструментарий текстового режима.
Python..........Язык программирования.
Rawrite..........Программа записи образов на дискеты.
SBM..........Smart Boot Manager.
SDL..........Библиотека мультимедиа.



› *Medit* – быстрый редактор для программистов, с примесью функций IDE.



› Нужен стойкий пароль? *JpasswordGenerator* вам его сделает.



› *Tor* обеспечит вашу анонимность онлайн, а *Tork* обеспечит к нему приятный интерфейс KDE.

Документ

КОНЦЕПЦИЯ
БАЗОВОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
В РОССИЙСКОЙ ФЕДЕРАЦИИ

документ подготовлен ОАО Линукс Инк., ЗАО Lynx BCC
в рамках деятельности рабочей группы при Минсвязи РФ (2003 г.)

Базовое программное обеспечение (БПО) включает в себя:

1. Операционные системы.
2. Программное обеспечение поддержки сетевой инфраструктуры – электронная почта, WWW, DNS, сервера баз данных и т.д.
3. Офисное или прикладное программное обеспечение (ПО).
4. Средства разработки.

1. ПРЕДПОСЫЛКИ СОЗДАНИЯ КОНЦЕПЦИИ

Технологические

Развитие Интернет сократило время на информационные коммуникации между людьми и обеспечило быстрый доступ к большим объемам информации. Этот факт и существенные инвестиции мировых лидеров отрасли информационно-коммуникационных технологий (ИКТ) обеспечили появление качественного ПО с открытыми кодами, которое можно использовать и модернизировать совершенно легально без каких-либо лицензионных платежей.

Правовые

В настоящее время на территории России большинство пользователей компьютеров, как в государственных учреждениях, так и в частном/индивидуальном секторе, пользуется нелицензионным ПО.

Экономические

В настоящее время вырученные денежные средства за ПО «уходят» за рубеж, стимулируя развитие отрасли ИКТ там, а не в России.

Исторические

Вследствие феномена свободного программного обеспечения – возможности воспользоваться на легальной основе через Интернет разработанным квалифицированными специалистами ПО – Россия именно сейчас получает возможность быстро, с наименьшими расходами создать собственную отрасль по производству ПО.

Политические

Происходит интеграция России в мировое сообщество, при этом БПО может обеспечить решение вопроса: как одновременно соответствовать общемировым информационным стандартам и обеспечивать информационную безопасность страны.

Социальные

Настоящая концепция позволит осуществить вовлечение в процесс популяризации информационных технологий школьников и студентов, которые через какое-то время станут определять информационный потенциал страны, как это происходило в развитых странах мира, когда основным каналом продвижения ИТ были университеты, колледжи, школы; малоимущие слои населения, что позволит им, получив дополнительные знания, применить их в быстрорастущем секторе рынка информационных услуг.

2. ОСНОВНЫЕ ТЕЗИСЫ

- создать отечественное БПО на основе имеющихся, доступных версий свободного программного обеспечения;

- создание российского БПО вызовет эффект «расходящихся кругов». На первом этапе создаётся отрасль, обеспечивающая функционирование БПО. Внедрение БПО в государственных учреждениях и использование такового в народном хозяйстве страны вызовет образование множества сервисных компаний, обслуживающих БПО, и компаний, ведущих собственные оригинальные разработки, в том числе ориентированные на специальные ведомственные или коммерческие задачи. Всё это стимулирует развитие внутреннего рынка ПО и даст толчок к развитию отрасли производства отечественного ПО в целом, повысит её конкурентоспособность на мировом рынке.

3. Генеральная цель

- Создать отечественную отрасль по производству и обслуживанию отечественного ПО на базе открытых программных продуктов.

4. Цели

- обеспечить государственные учреждения лицензионным БПО;
- за счёт внедрения БПО в государственных учреждениях обеспечить информационную безопасность страны;
- развить отечественную отрасль ИКТ в части производства ПО;
- внедрить современные бизнес-модели и стандарты разработки ПО;
- интегрировать Россию в мировое сообщество разработчиков ПО, выйти на мировой рынок, удержать в стране интеллектуальный потенциал государства;
- создать дополнительный рынок труда;
- решить проблему занятости и проведения досуга молодежи;
- создать и развить инфраструктуру оказания услуг по поддержке ПО;
- сформировать рынок оказания ИТ-услуг, как для отечественного потребителя, так и для иностранных компаний;
- снизить уровень использования нелицензионного ПО, что будет способствовать повышению статуса государства.

5. Задачи и мероприятия

- разработать набор стандартных пакетов безопасного ПО;
- разработать пакет технической и пользовательской документации;
- создать программу внедрения БПО в государственных учреждениях;
- обеспечить посредством обучения достаточное количество квалифицированных сотрудников-системных администраторов в государственных учреждениях;
- обеспечить обучение сотрудников государственных учреждений – пользователей;
- обеспечить инфраструктуру для работы организаций – разработчиков ПО.

6. Ожидаемые эффекты

Экономический

- достижение глобальной экономии государственных (бюджетных) средств на поддержание БПО в государственных учреждениях – за счёт резкого снижения затрат на приобретение лицензий у западных производителей;
- высвобождение средств на развитие собственной ИКТ индустрии, что обеспечит конкурентоспособность отечественных разработок;
- смещение акцентов в народном хозяйстве России с развития сырьевых отраслей на развитие отрасли информационных технологий;
- дальнейшее развитие сферы услуг в области ИТ;
- повышение экономической прозрачности деятельности предприятий госсектора;
- в среднесрочной перспективе – увеличение объёма налогов, собираемых не только в отечественной ИКТ отрасли, но и в целом в народном хозяйстве; развитие данного направления ИКТ отрасли способствует образованию критической массы квалифицированных специалистов, обеспечивающих внедрение современных информационных технологий на предприятиях;
- кардинальное снижение уровня пиратства позволит России стать полноценным членом мирового сообщества, в частности будет способствовать вступлению в ВТО.

Социальный

- создание новых рабочих мест;
- наращивание интеллектуального потенциала в России – воспитание большого количества высококвалифицированных программистов;
- решение проблемы занятости и проведения досуга молодежи.

Военный

- обеспечение безопасности государства – полный, реальный контроль над качеством используемого в государственных учреждениях, в том числе силовых структурах, программного обеспечения.

Технологический

- возможность быстрой технологической адаптации и модернизации БПО;
- более эффективная работа БПО в государственных учреждениях благодаря техническим особенностям БПО: уникальной масштабируемости, отсутствию компьютерных вирусов (в общепринятом понимании), широкому диапазону использования БПО – от встраиваемых портативных систем до суперкомпьютеров;
- предоставление унифицированных решений на различных уровнях (предприятие, регион, федерация) с использованием всех преимуществ, даваемых современными информационными технологиями;
- возможность устойчивой работы и обеспечение удаленного доступа к объединенным информационным государственным ресурсам, в том числе в рамках программы «Электронная Россия»;
- возможность влияния на мировой ИТ рынок.

Развитие науки и образования:

- активное вовлечение научных и учебных заведений всей страны в процесс разработок, что позволяет реализовать масштабные общероссийские проекты;
- повышение качества образования, как в средних, так и в высших и специальных учебных заведениях

Реализация целевой государственной программы, направленной на создание отечественной отрасли, которая обеспечивает разработку и сервисное сопровождение российского **Базового программного обеспечения**, будет способствовать быстрому росту информационной и экономической независимости и, следовательно, безопасности России.

Экономическое обоснование использования разработок с открытым кодом

Использование свободного ПО – наиболее быстрый и эффективный путь внедрения в общемировой рынок ПО. Индустрия развития информационных технологий – на данный момент наиболее быстро развивающийся сектор мировой экономики. Состояние рынка в этой области меняется очень быстро, и у России есть реальный шанс занять в нем одну из лидирующих позиций, особенно в сфере разработки ПО. Этому способствует наличие у нас в стране большого числа высококвалифицированных специалистов в данной области.

В настоящий момент на мировом рынке ПО сложилась уникальная ситуация – США более не является монополистом в области передовых программных разработок. Все больше программных проектов, как коммерческих, так и свободных, создается и развивается за пределами США. Linux был рожден в Финляндии, GNOME – Бразилия, KDE и SAP – Германия, Compiere ERP + CRM – Франция. Большинство разработок делаются с привлечением специалистов из самых разных стран мира. В России огромный потенциал разработчиков. В большинстве международных проектов по разработке свободного ПО участвуют российские разработчики.

Главные моменты при выборе ПО для государственных органов: во–первых – обеспечение информационной безопасности страны, во-вторых – деньги должны оставаться в России. Вместо обязательных платежей за лицензии государство сможет вкладывать средства в сферу образования, в создание рабочих мест для отечественных специалистов. При государственном финансировании разработок следование открытым стандартам (например, POSIX) и открытость кода должны быть обязательными требованиями.

Известны две схемы развития индустрии отечественного ПО:

1. Офшорное программирование.
2. Разработка собственных продуктов и распространение их на общемировом рынке товаров и услуг.

До сих пор развитие индустрии ПО в России в большинстве случаев идет по первому пути, когда результаты разработок оказываются собственностью зарубежных компаний и через какое-то время поступают на российский рынок как импортируемый продукт. Второй путь гораздо перспективнее, но требует финансирования фундаментальной науки (там отрабатываются новые идеи) и инвестиций в разработку конечного продукта (в том числе и государственных).

Крайне важным для развития индустрии является правильный выбор политики государства в области поддержки открытых стандартов. Уже сейчас в стране существует ряд достаточно успешно работающих фирм-разработчиков программного обеспечения, создающих собственные программные продукты. К сожалению, большинство из них в качестве базового программного уровня используют закрытые коммерческие системы иностранного производства (например, Microsoft Windows). Это не способствует повышению экономической безопасности страны, ставя ее в прямую зависимость от интересов фирм производителей и государственной политики тех стран, откуда производится экспорт таких систем.

Возможные пути развития новых технологий – корпоративные разработки, государственное финансирование (гранты, вузы, специализированные институты), деятельность независимых разработчиков. Мировой опыт показывает: большинство инновационных технологических разработок рождается в вузах, зачастую как побочный результат фундаментальных исследований. Существенную долю в разработку нового ПО вносят студенты, занимаясь этим в свободное время и повышая таким образом свой статус. Коммерческие структуры редко тратят деньги на действительно инновационные исследования, обычно они используют уже готовые результаты научных исследований.

Развитие ПО очень похоже на научный процесс. Общая открытость его только ускоряет. Лучшая черта открытого ПО – расширяемость. Типичная ситуация – 80% требуемой функциональности уже есть в каких-то продуктах и требуется сравнительно небольшая их доработка для решения новых задач.

Открытое ПО дает конкурентные преимущества даже в производстве коммерческого ПО. Большинство ведущих ИТ корпораций всё шире используют открытые программные продукты, разработанные мировым сообществом разработчиков, или же открывают исходные тексты своих продуктов. Покупатель больше не хочет платить за отдельные компоненты, предпочитая целостные решения и сервис. При этом достигается большая независимость от производителей и сохраняется полный контроль со стороны покупателя над приобретаемыми продуктами.

Вопросы безопасности

Только использование ПО с открытыми кодами может дать **гарантию госу**дарственной безопасности в сфере информационных технологий.

Только при таком подходе возможны:

- полный анализ содержимого исходных кодов;
- оперативное приведение их в полное соответствие с государственными правовыми и нормативными требованиями;
- контроль за ходом технологического процесса на всех этапах от разработки до окончательного внедрения.

При таком подходе принципиально невозможно использование со стороны поставщика или изготовителя ПО так называемых «программных закладок», способных производить неподконтрольную передачу внутренней информации или преднамеренно нарушать работу вычислительных комплексов.

Открытое ПО предоставляет России реальную независимость как от коммерческой политики фирм производителей, так и от ограничений, накладываемых на государственном уровне странами экспортерами программных продуктов.

Правовые вопросы

Кардинально решить проблему пиратства можно лишь при условии широкого внедрения свободно распространяемого ПО с открытым кодом. Это не требует внешних лицензионных выплат и позволяет продолжать пользоваться привычным набором ПО абсолютно легально.

Существуют различные варианты свободных лицензий и различные варианты их использования. Наибольшее распространение получили два типа свободных лицензий. Условно их можно охарактеризовать следующим образом:

1. Универсальная общественная лицензия GNU (GNU General Public License, сокращенно GPL). Разрешает свободное использование исходных кодов программных продуктов, попадающих под ее область действия (в том числе и в коммерческих целях), и внесение в них любых изменений. В случае использования их в своих разработках разработчик обязуется в дальнейшем предоставлять свои исходные коды по первому требованию.
2. Программная лицензия университета Беркли (Berkeley Software Distribution, сокращенно BSD). Так же предоставляет право неограниченного использования в сторонних разработках, но, в отличие от GPL, позволяет в дальнейшем сделать продукт закрытым.

Другие свободные лицензии в той или иной степени дополняют изложенные выше требования одной или другой лицензии.

Использование в качестве базового открытого программного обеспечения никоим образом не сковывает производителей программных продуктов. Даже при использовании программных продуктов под лицензией GPL производитель не обязательно должен открывать свой код (например, в том случае, когда используются разделяемые библиотеки или создаются подгружаемые модули ядра ОС).

Международный опыт разработки/внедрения открытых технологий

В Индии и Китае разрабатываются экономические программы для стимулирования перехода различных отраслей на открытые технологии, а также сделаны заявления об использовании Linux в качестве официальной государственной ОС. Сингапур, Тайвань, Германия выразили намерение внедрять серверные решения на базе Linux, мотивируя свой выбор прежде всего соображениями существенной экономии средств.

В мае 2003 г. муниципалитет Мюнхена (Германия) принял решение о полномасштабной миграции с платформы Windows на Linux для 14,000 персональных компьютеров. Другие города Германии также рассматривают вариант перехода на открытые операционные системы и офисные пакеты. В любом случае они выигрывают, поскольку конкуренты (Microsoft) вынуждены в таких случаях предлагать существенные скидки на свои продукты, как это произошло в тендере ПО для города Франкфурт.

Английское правительство сумело добиться экономии в 150 миллионов долларов при заключении аналогичного контракта на три года. Проект положения об использовании лицензий open-source в ПО, разрабатываемом по заказу правительства, служит частью более широкого положения об open-source, которое прошлым летом было опубликовано Управлением снабжения правительства (OGC) Великобритании. Согласно этому документу во всех будущих ИТ-разработках, для которых важна совместимость, должны использоваться только продукты, поддерживающие открытые стандарты и спецификации. Более того, он требует выполнения политического документа Европейской комиссии, предписывающего изучение возможности следования курсом open-source во всех финансируемых государством исследованиях и разработках в области ПО.

Установка свободного ПО (Linux, GNOME и др.) на 80,000 школьных компьютерах в провинции Extremadura (Испания) позволила сэкономить около трети выделенного бюджета (порядка 18 миллионов евро). Там же была развернута программа по созданию 33 общедоступных компьютерных центров на базе открытого ПО для привлечения населения к использованию современных средств коммуникации (например, e-mail). В программе приняли участие местные жители различных социальных групп и возрастов. Многие из них прежде никогда не использовали компьютеры. Старейшему зарегистрированному пользователю было 99 лет.

В мире научных разработок открытые технологии являются стандартом де факто. Именно благодаря разработкам открытых технологий мир получил глобальную сеть Интернет. Широко применяется открытое ПО в сферах, связанных с интенсивными вычислениями – ядерная физика, геофизика и др. Это позволяет создавать мощные вычислительные комплексы с использованием дешевой аппаратной базы.

Большой интерес к открытому и свободному ПО проявляют и силовые государственные структуры. Так Агенство Национальной Безопасности США (NSA) создает свой вариант Linux-системы повышенной степени защищенности SELinux. Большой интерес к использованию открытого ПО в своих внутренних структурах проявляют так же ЦРУ, ФБР и министерство обороны США.

Серьезно меняется подход к использованию свободного ПО и в коммерческой сфере. Компании Merill Lynch, Verizon Communications, Amazon.com сэкономили миллионы долларов на лицензиях и стоимости поддержки, уменьшив общую стоимость владения своей ИТ-инфраструктуры вследствие перехода на использование ПО с открытым кодом.

Наблюдается стремительный рост поддержки открытых технологий ведущими коммерческими производителями ПО и компьютерного оборудования: IBM, Hewlet Packard, Sun Microsystems, Oracle, Intel и др.

Основные критерии оценки программного обеспечения

1. Соответствие открытым стандартам. Снижает степень зависимости от конкретных производителей и поставщиков услуг. Упрощает построение единых программно-аппаратных комплексов из отдельных независимых компонент. Облегчает замену одних компонент на другие (например, вследствие развития более современных технологий).
2. Переносимость программного обеспечения. Облегчает процесс миграции наработанного ПО на более совершенные аппаратные платформы. Закладывается прочный базис для создания и развития отечественного производства высокотехнологичных аппаратных компонент и комплексов.
3. Открытость исходных кодов. Дает возможность анализа, внесения изменений для получения более широкой функциональности, возможность повторного использования ранее созданных компонент.

Бесплатность подавляющего числа открытых программных продуктов может существенно снизить накладные расходы при грядущем кардинальном перевооружении информационной инфраструктуры страны, необходимом для полноправного вхождения страны в международный рынок товаров и услуг. Было бы разумно воспользоваться общемировой тенденцией смещения рынка из области продажи изолированных программных продуктов (зачастую предоставляемых теперь бесплатно) в область оказания интеграционных и сервисных услуг.

LINUX FORMAT
Главное в мире Linux

Журнал зарегистрирован Федеральной службой по надзору за соблюдением законодательства в сфере массовых коммуникаций и охране культурного наследия
ПИ № ФС77-21973 от 14 сентября 2005 года
Выходит ежемесячно. Тираж 5000 экз.

РЕДАКЦИЯ РУССКОЯЗЫЧНОЙ ВЕРСИИ:
ГЛАВНЫЙ РЕДАКТОР
Валентин Синицын info@linuxformat.ru
Литературные редакторы
Родион Водейко, Елена Толстякова, Иван Мищенко
Переводчики
Александр Бикмеев, Светлана Кривошеина, Александр Кузьменков, Алексей Опарин, Валентин Развозжаев, Сергей Супрунов, Александр Черных, Юлия Шабунио
Допечатная подготовка
Мария Пучкова, Родион Водейко
Креативный директор
Станислав Медведев
Технический директор
Денис Филиппов
Директор по рекламе
Денис Игнатов +7 812 965 7236 advert@linuxformat.ru
Заместитель генерального директора
Софья Виниченко
Генеральный директор
Павел Фролов
УЧРЕДИТЕЛИ
частные лица
ИЗДАТЕЛИ
Станислав Медведев, Павел Фролов

Отпечатано в типографии «Текст», ООО «ППК «Текст»
188680, Ленинградская область, Всеволожский район, Колтуши, д.32
Заказ _____

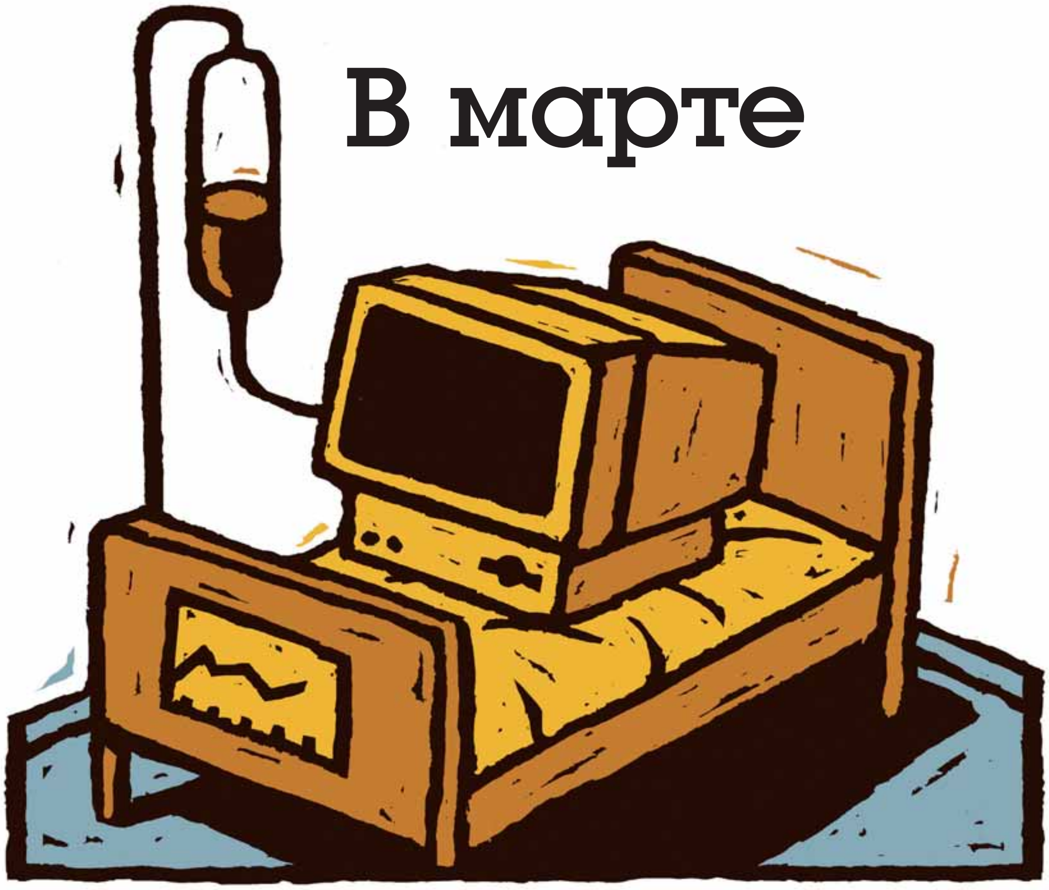
Пре–пресс: *d.r.i.v.e-group*

РЕДАКЦИЯ АНГЛОЯЗЫЧНОЙ ВЕРСИИ:
Редактор Ник Вейч (Nick Veitch) nick.veitch@futurenet.co.uk
Заместитель редактора Пол Хадсон (Paul Hudson) paul.hudson@futurenet.co.uk
Старший художественный редактор Мартин Парфитт (Martin Parfitt) mparfitt @ futurenet.co.uk
Художественный редактор Эфрейн Эрнандес-Мендоса (Efrain Hernandez-Mendoza) efrain.hernandez-mendoza@futurenet.co.uk
Новостной редактор Майк Сондерс (Mike Saunders) mike.saunders@futurenet.co.uk
Литературный редактор
Ребекка Смолли (Rebecca Smalley) rebecca.smalley@futurenet.co.uk
Штатный автор
Грэм Моррисон (Graham Morrison) graham.morrison@futurenet.co.uk
Ассистент по выпуску
Эндрю Грегори (Andrew Gregory) andrew.gregory@futurenet.co.uk

Авторы
Ладислав Боднар (Ladislav Bodnar), Нейл Ботвик (Neil Bothwick), Д-р Крис Браун (Dr. Chris Brown), Энди Ченнел (Andy Channelle), Ричард Драммонд (Richard Drummond), Стефан Лукас (Stephan Lucas), Евгений Балдин, Андрей Боровский, Андрей Прахов, Петр Семилетов, Александр Супрунов, Алексей Федорчук, Александр Бабаев, Илья Шпаньков
Художественные ассистенты: Анна Фишер (Anna Fisher), Дамиан МакГи (Damian McGee), Энди Онстед (Andy Ounsted), Эмит Пэтел (Amit Patel)
Фотографии: Джейсон Каплан (Jason Kaplan)
Иллюстрации: Нейл Бартлет (Neil Bartlett), Пол Блехфорд (Paul Blachford), Elly Walton Illustrations, Крис Винн (Chris Winn)

КОНТАКТНАЯ ИНФОРМАЦИЯ
UK: Linux Format, 30 Monmouth Street, Bath BA1 2BW
Tel 01225 442244 Email: linuxformat@futurenet.co.uk
РОССИЯ:
Санкт-Петербург (редакция): ул. Гончарная, 23, офис 54, телефон: (812) 717-00-37
Представительство в Москве:
пр.Мира, 161, телефон +7(495) 799-18-63, +7(495)136-88-45
Email: info@linuxformat.ru, Web: www.linuxformat.ru

В марте



Почини сам!

Устали от загадок, которые подкидывает Linux? Мы изучили типовые проблемы, возникающие у пользователей и расскажем вам, как решить их!

Супер Apache

Ваш сервер умеет многое – в следующий раз мы рассмотрим, какие модули существуют для Apache и как их использовать

Что за штука... OpenID ?

Децентрализованная цифровая личность – что это значит?



Дамиан Конвей

Поболтаем о фичах Perl 6.

Внимание! Содержание следующих выпусков может изменяться без уведомления